Reducing power consumption of mobile thin client devices

F. Azmat Ali[1], P. Simoens[1], B. Vankeirsbilck[1], L. Deboosere[1], B. Dhoedt[1], P. Demeester[1], R. Torrea-Duran[2], C. Desset[2]

[1]IBBT-IBCN, Department of Information Technology, Ghent University, Gent, Belgium; [2]IMEC, Leuven, Belgium

E-mail: [1]farhan.azmatali@intec.ugent.be, [2]torrea@imec.be

Abstract: **The thin client computing paradigm shifts processing capabilities from the device to the network. Applications are executed on network servers and the terminal functionality is limited to capturing user input and rendering the display updates. Since the amount of processing by the terminal is reduced, thin clients are potentially energy efficient devices, making them highly appealing to mobile users. However, intensive network communication is required to convey user input and display updates between server and terminal. The increased power consumption of the device radio platform might undo or even exceed the savings achieved by the reduction in local processing. In this paper, two power-saving strategies for thin client are presented: adaptive content encoding and improved wireless link layer control algorithm. The strategies have been experimentally validated in three scenarios. This work was done in the scope of the European FP7 MobiThin project, focused on the adaption of wireless and mobile thin client computing.**

Keywords: thin client, power consumption, wireless link layer control, adaptive content encoding

1 INTRODUCTION

A thin client heavily relies on a remote server for processing activities. It merely acts as an interface, conveying input and output between the user and the remote server, rather than executing the applications locally. The required client functionality is limited to capturing user input and rendering display updates received from the remote server. The communication is established through a remote display protocol.

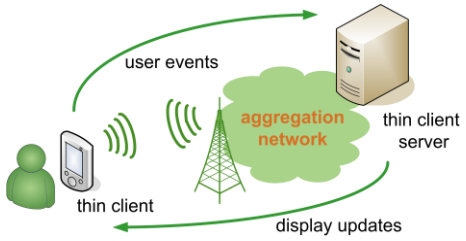


Figure 1 - User input is sent to the application server. Resulting display updates are sent back to the client.

The concept enables the consumption of even the most demanding applications and services on resource constrained terminals. This is of particular interest in the context of mobile devices, where users are nowadays often obliged to fall back on a restricted version of the applications, tailored to the specifics of the underlying mobile phone operation system. Another major advantage of mobile thin client computing is that the reduction in required terminal processing hardware allows to make the mobile device more lightweight and potentially more energy efficient.

The thin client paradigm trades local processing for network communication. In a mobile device context, the terminal will most probably be connected to the server over a wireless interface, such as WiFi, UMTS or LTE. The power required by the wireless network interface card to transmit and receive remote display protocol traffic might undo, and eventually exceed the power savings achieved by the reduction in processing on the terminal.

This paper contributes to an optimization of wireless thin client devices. It carries out an analysis of the power consumption of the wireless platform due to the remote display protocol traffic and explores two ways to optimize the power consumption, located at different layers of the communication stack. The first strategy is an improvement of the wireless link layer control algorithm on the device, by taking into account the current state of the physical radio interface. At a higher layer in the communication stack, the presentation layer, terminal power savings are achieved through a more efficient content encoding by the application server, thereby reducing the required bandwidth and the terminal decoding effort.

The reported research has been conducted in the scope of the FP7-funded MobiThin project [1], addressing important blockers of the wide adoption of the wireless thin client computing paradigm. In addition to the design of a suitable framework for thin client service delivery and provisioning, research is conducted into remote display protocols. This includes the development of a protocol adapting to the application content and wireless network characteristics. Specific attention is paid to minimize the induced terminal power consumption.

The remainder of this paper is structured as follows. A survey of related work on power consumption of thin clients is presented in section 2. Both power optimization strategies are elaborated on in section 3. Section 4 details the experimental setting in which the results of section 5

Corresponding author: Farhan Azmat Ali, IBBT-IBCN, Deparment of Information Technology, Ghent University, Gaston Crommenlaan 8 bus 201, 9050 Gent, +32 9 3314900, farhan.azmatali@intec.ugent.be

have been achieved. The main conclusions are set out in section 6.

2 RELATED WORK

The power efficiency of thin clients, compared to desktop PCs, was previously studied in [2]. The authors demonstrate how some thin client devices use up to 85 % less power than their PC rivals in a real world environment. A more generic viewpoint is taken in [3], where the authors study the environmental impact of thin clients compared to desktop PCs, taking into account the production, distribution and operational phase. In previous work, we studied the impact of the increased power consumption of application server farms (server powering, cooling) on the power efficiency of the thin client use case, compared to the classical desktop case [4],[5].

In [6], we carried out a detailed analysis of the power consumption due to thin client protocol traffic. Some guidelines for wireless link layer optimization were formulated, which are validated in the current paper. Technical details on the adaptive content encoding mechanism, which is the second power optimization strategy under study in this paper, are given in [11]. While the previous paper focused solely on the bandwidth and QoS optimization, the present paper studies the power efficiency of this approach.

3 POWER OPTIMIZATION

This section describes two different power optimization strategies for a thin client environment. As outlined above, the first strategy is located at the wireless link layer, while the second is located at the upper layer of the communication stack: the presentation layer.

3.1 Wireless link layer control

The radio platform of a device can be in 4 different power states: sleep, idle, send and receive mode. These states are differentiated by the activation of certain platform components. In idle state the terminal can detect incoming frames because the front-end is activated, while in sleep mode all components are turned off and no incoming frames are detected. The send and receive state are the most power consuming and correspond to the terminal state of transmitting or receiving data. The total radio platform energy consumption depends not only on the active components in a given radio power state, but also on the time they remain in that state. This is steered by the wireless link layer control algorithm.

As the power amplifier is the most power consuming component, most reference control algorithms adopt the maximum-throughput strategy, aiming to transmit the data as fast as possible to have some time spent in idle or sleep mode afterwards. In the remainder of this paper, we refer to this algorithm as Reference Solution (RS).

However, the RS leads to a maximum activation of the power amplifier. As the power amplifier consumption does not scale linearly with the transmission rate, transmitting at a lower rate and thus a lower activation of the power amplifier might turn out to be more energy efficient, although we spend more time in the sending state. In [6], an improved control algorithm was presented, that takes into account models of the platform power consumption and the current radio channel statistics. Based on a predefined database with optimized configurations in function of the current channel conditions and the data rata requirements, link layer parameters are tuned, such as modulation and code rate, as well as power amplifier settings, such as the signal output power. As the algorithm works cross-layer between the wireless link and the physical layer, it will be referred to in this paper as XCTRL.

3.2 Adaptive content encoding

Current remote display protocols such as VNC-RFB [7], Microsoft's RDP [8] and Citrix' ICA [9] are optimized for rather static displays of office applications, such as a text editor or a spreadsheet. The transport of multimedia data over a remote display protocol is very inefficient, leading to high bandwidth requirements [10]. Video codecs such as MPEG-2 are optimized to handle fast moving images in a bandwidth efficient manner. We have developed an adaptive encoding infrastructure, depicted in Figure 2.

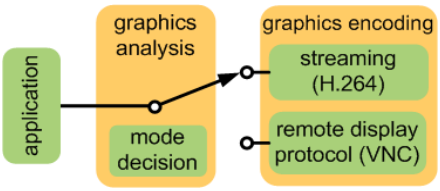


Figure 2 - Adaptive encoding. The amount of motion in the graphics is analyzed and the best transmission mode is selected.

The amount of motion in the graphics is analyzed. For static displays, the VNC mode is selected and for high-motion graphics the streaming (H.264) mode is applied. In [11], we have detailed this architecture and demonstrated the gains in bandwidth. While the streaming mode can offer smooth multimedia graphics, it was shown that the decoding of H.264 requires more CPU cycles than decoding VNC, advocating the overhead hybrid mechanism as it allows to switch back to VNC for low-motion scenarios. In the current paper, we investigate the power consumption of both modes and study if the wireless platform energy savings due to the bandwidth reduction of the streaming mode is not undone by the increased decoding complexity.

4 EXPERIMENTAL SETTING

4.1 Test scenarios

To isolate the effects of both mechanisms described in the previous section, different test scenarios were elected to validate both mechanisms.

4.1.1 Wireless link layer control algorithm

In order to asses the specific benefits of the wireless link layer control algorithm, the content encoding should be fixed, in our case we chose to fix it to VNC-RFB. As

VNC is designed for office related work, the elected test scenarios cover text editing and browsing, while multimedia was excluded to avoid biasing the test results [10]. The main differentiator is the amount of generated traffic. The three scenarios listed below are in increasing order of required downstream bandwidth.

- Text editing: the user opens a text editor (OpenOffice Writer), inserts a title and subtitle and types half a page of text. Afterwards, a picture is inserted and a few more sentences are typed, before closing the text editor.
- Static browsing: the user navigates through the IceWeasel browser to different websites containing rather static content. Successively, the user logs in to a GMail account, reads and replies to an e-mail, and reads a few articles on the BBC news website. As the news website is updated frequently, it was made available offline to keep the content exactly the same over the different measurement iterations.
- Dynamic browsing: the user navigates through the IceWeasel browser to different websites, containing more dynamic content like JavaScript or Flash. Visited websites include www.creaktif.com, www.cafesonique.com, www.youtube.com

4.1.2 Adaptive encoding

The scenarios described above are not suited to test the adaptive encoding mechanism because no switching between the encoding modes would occur. Therefore, two new test scenarios were selected, specifically oriented to one of both encoding modes. More specifically, video was included to test the streaming mode.

- Static: a text editor (OpenOffice Writer) is started through the menu bar of the desktop. A text is typed, this text is then selected and aligned by the "justify" command. A figure is inserted, followed by a caption of two lines of text. At the end, the user scrolls back to the top and closes the text editor.
- Dynamic: a video file is played with the VLC media player in full screen mode. The video repeats in an infinite loop.

4.2 Measurement set-up

The testbed is depicted in Figure 3. It comprises a VNC server and client, connected via an internal fixed network to the 802.11 link simulator. At the terminal, the VNC viewer translates the user events to VNC protocol messages, which are then relayed to the VNC server through the simulator. The server encodes the corresponding graphical updates in VNC protocol messages to be returned to the terminal.

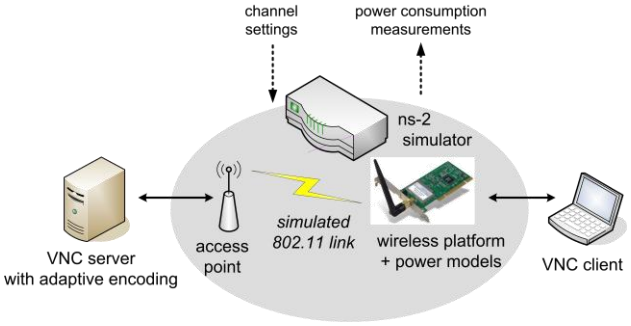


Figure 3 - The testbed comprises a VNC server and client, connected by a simulated 802.11 link. The NS-2 simulator models the wireless channel transmission as well as the power consumption of the wireless platform on both access point and terminal.

The 802.11 link simulator is completely transparent to the server and the terminal. Implemented in NS-2, its functionality is twofold. First of all, it implements a MAC and PHY stack to mimic data transmission between access point and terminal over 802.11. Different channel parameters can be configured: path loss, coherence time and terminal speed.

The second functionality of the simulator is the assessment of the power consumption. For each of the 4 modes of the wireless platform, models of the power consumption of the various parts of a wireless platform of both the access point and the user terminal were implemented in the simulator. The simulator keeps track of how much power is consumed by the wireless platform in every mode, at both the access point and the user terminal.

5 EXPERIMENTAL RESULTS

5.1 Wireless link layer control algorithm

The power consumed by the wireless platform was measured for three scenarios described in section 4.1.1, using the testbed depicted in Figure 3. For every scenario, a series of user events was recorded by an in-house developed VNC replay tool, ensuring that exactly the same application layer traffic is generated when repeating the same scenario for different wireless channel parameter configurations.

The channel coherence time is set to 0.1 s, 1 s and 10 s. Experiments are performed for both static and moving terminals. For a static terminal, located at a fixed distance from the base station, the average path loss is varied from 60 dB to 90 dB. For a moving terminal, the terminal is located at the start at a distance of 10 m from the base station and moves away from base station with fluctuating and increasing path loss. At the end of trajectory, the terminal is located at 40 m from the base station. The movement speed of the terminal is set to 1 m/s, 3 m/s and 5 m/s.

From all channel condition parameters, the channel path loss seems to have the largest influence on the power consumption, while the impact of the channel coherence time proves to be rather minimal. Figure 4 demonstrates

the influence of the channel condition on the power consumption.

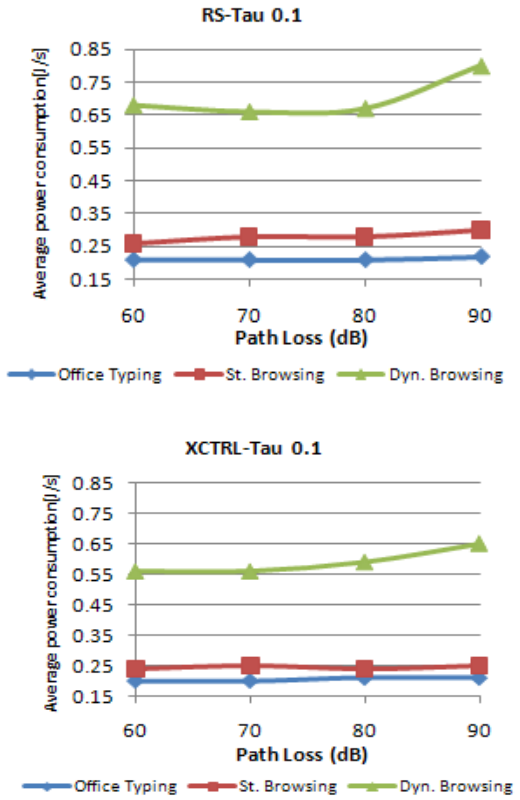


Figure 4 - Average power consumption for different scenarios with static terminal

For both RS and XCTRL, the power consumption increases with the path loss. A degraded channel leads to an increase in the number of retransmissions and requires using less power efficient data modulations. This increases the time and energy spent in send and receive mode. The XCTRL algorithm takes into account the current channel conditions and adapts its transmission strategy, as previously described in section 3.1. The gains increase when there is more downstream traffic, as the terminal has to send more acknowledgements (ACKs) and accordingly can go more into sleep mode immediately after sending these ACKs.

Table 1 and Table 2 compare the energy consumption in all three scenarios when the RS or the XCTRL algorithm is applied.

Table 1 - Comparison of the power consumption for channel path loss of 60 dB, coherence time 0.1 s and a static terminal

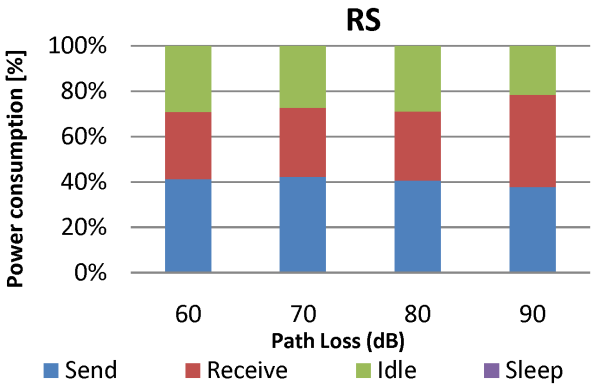
Scenarios	RS [J/s]	XCTRL [J/s]	Gain [%]
Office typing	0.21	0.20	5
Static Browsing	0.26	0.24	8
Dynamic browsing	0.68	0.55	23

Table 2 - Comparison RS and XCTRL for a channel path loss of 60 dB, coherence time 0.1 s and a moving terminal at 1 m/s

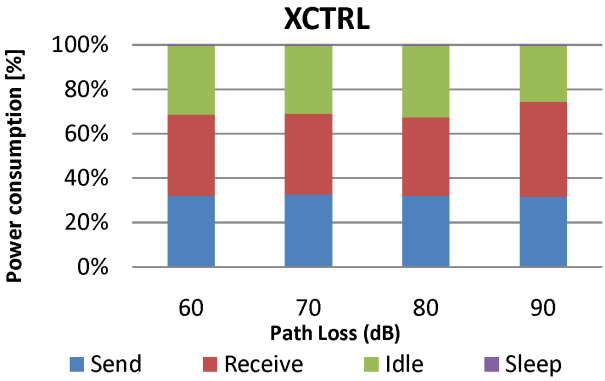
Scenarios	RS[J/s]	XCTRL[J/s]	Gain[%]
Office typing	0.21	0.20	5
Static Browsing	0.27	0.25	8
Dynamic browsing	0.72	0.62	16

The total power consumption is the sum of the energy consumed in send, receive, sleep and idle mode. With current experimental conditions, having no background traffic in the network, most of the power is consumed in idle mode. For low traffic scenario, e.g. text editing, this happens due to the fact that there is only small amount of data to send or receive. Consequently, 90 % of total power is consumed in idle mode. For high traffic scenario, e.g. dynamic browsing, idle mode consumes almost half of the total power consumption.

Figure 5 shows a breakdown of the power consumption by thin client in the different modes for dynamic browsing scenario. This is the scenario with the most downstream traffic. The other scenarios have not been included due to space constraints and because the conclusions are largely similar.



(a) Reference solution - channel coherence time = 0.1s



(b) XCTRL - channel coherence time = 0.1s

Figure 5 - Power consumption in different modes for the dynamic browsing scenario

From Figure 5, it is clear that the XCTRL algorithm is able to reduce the relative (and absolute) power consumed

in the send state, because of its more efficient transmission strategy explained before.

The energy spent in the receive state is largely dependent on the amount of bytes generated by the application server. Consequently, no significant wireless link layer strategy at the client can be expected here, rather some efficiency improvements should be done when encoding the data. The adaptive encoding mechanism is an example of such an optimization and is explained in the next section.

Besides limiting the time spent in send and receive state, energy gains could be achieved by putting the terminal more in sleep mode and less in idle mode. Currently, the terminal cannot go into sleep mode, as it might miss any relevant data. If the application and wireless platform could agree on data-free periods between the client and base station, the terminal could be put in sleep mode, as it can be sure not to miss any relevant data. As this mechanism would affect the throughput and increase additional delays, care should be taken to maintain a sufficient user Quality of Service, e.g. the sleep interval should not significantly degrade the user interactivity experience.

5.2 Adaptive content encoding

Commercially available hardware devices were used to test the adaptive content encoding mechanism, in order to include the power consumption of the CPU when decoding and rendering the content. Hardware specifications are shown in Table 3. The client is connected to the server via a WiFi access point.

Table 3 - Hardware characteristics

Server	**Thin Client**
Intel QuadCore 2.73 GHz	Asus EEE pc Intel Celeron M
2 GB RAM	512 MB RAM
1 Gbps NIC	WiFi
Linux Ubuntu	Linux Debian
Hybrid VNC server	Hybrid VNC client

The energy measurements are performed by removing the battery from the device and measuring the current drawn from and voltage over the DC side of the adapter. Both current and voltage are simultaneously measured with a separate Velleman DVM345DI Digital Multimeter. One sample is taken per second. Every experimental run lasted one hour.

Table 4 shows the average energy consumed by both encoding mechanism, while executing the scenarios of section 4.1.2. The switching algorithm was disabled, and so either the VNC or the streaming encoding mode was applied during a single experiment. From an energy point-of-view, the table shows that VNC, is power efficient for office-type of applications but does not cope well with video-type of applications. 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, 4% of the consumed energy can be saved when transferring dynamic content in streaming instead of VNC mode. For static content, VNC is the better option because when streaming mode would be used, 1.6 % more energy would be consumed. Although the differences in energy consumption for both modes are relatively small, we remind the reader that the adaptive content encoding mechanisms was primarily designed to combine bandwidth efficiency and optimized QoS [11]. These results confirm that, for dynamic content, the more complex decoding of the streaming mode is still more power efficient than using VNC.

Table 4: Average energy consumption of streaming and VNC for both scenarios

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

Figure 6 depicts in more detail the momentary consumed power related to the encoding applied for the graphical updates. In this one-hour experiment, 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, again followed by 20 minutes of looping the static scenario. The full line shows the energy consumed by the adaptive encoding protocol. The dashed line shows the average energy that would be consumed if in all cases the "inappropriate" encoding would be applied, following Table 4: streaming static content and encoding dynamic content using VNC.

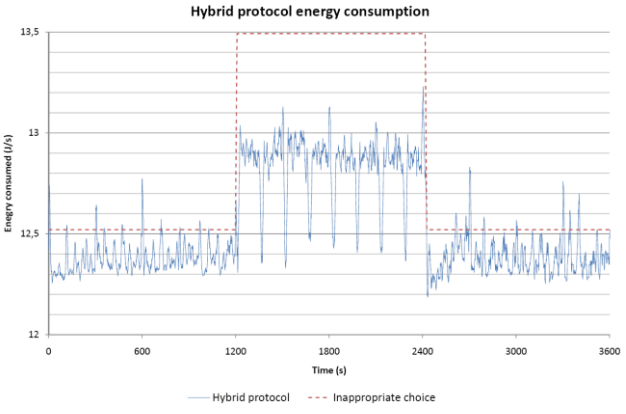


Figure 6 – Energy consumption of the Asus EEE PC.

Besides the already demonstrated bandwidth efficiency and QoS improvements [11], this experiment proves that the adaptive encoding mechanism chooses the less power consuming mode as well.

The spikes in the graph can be explained by a temporary switch in encoding mode. The mode decision 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 6, 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.

6 CONCLUSIONS

Significant terminal power savings can be achieved by delegating the application processing to an application server in the network. An intensive network communication is required to exchange user input and display updates between terminal and server. On wireless devices, the incurred power consumption of the wireless platform might undo or even exceed the power savings realized by the reduction in terminal processing.

In this paper, two strategies were presented to limit the power consumption on thin clients. By taking into account physical layer parameters to tune the wireless link layer transmission settings, energy savings up to 23 % were achieved. The second strategy aims for an optimization of the content encoding: static encoding is encoded through the VNC protocol, while more dynamic multimedia content is encoded through video streaming. It was shown that this approach results in more efficient power consumption, along with a reduction in bandwidth and an optimization of QoS.

Future work in this area will merge both tracks. Through a cross-layer framework, the wireless link layer control algorithm will communicate with the application and encoding layer. Feedback can be given on the available throughput of the wireless channel to tune the content encoding parameters.

Acknowledgment

Part of the research leading to these results was done for the MobiThin project and has received funding from the European Community's Seventh Framework (FP7/2007-2013) under grant agreement no. 216946. P. Simoens is funded through Ph.D. grant of the Fund for Scientific Research, Flanders (FWO-Vlaanderen). B. Vankeirsbilck and L. Deboosere are funded through Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

[1] MobiThin. http://www.mobithin.eu
[2] S. Greenberg et al., "Desktop energy consumption, a comparison of thin clients and pcs". WYSE White Paper.
[3] E. Weidner, "Environmental comparison of the relevance of pc and thin client desktop equipment for the climate, 2008". Fraunhofer Institute, April 2008.
[4] W. Vereecken et al., "Energy efficiency in telecommunication networks". Proc. Of European Conference on Networks & Optical Communications (NOC), July 2008, Krems, Austria.
[5] W. Vereecken, "Thin client power efficiency". Proc. Of KEIO and Ghent University 2nd G-COE Joint workshop for future network, September 2008, Ghent, Belgium.
[6] P. Simoens et al.,"Characterization of power consumption in thin clients due to protocol data transmission over IEEE 802.11". Proc. of 7th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2009, Seoul, South-Korea.
[7] T. Richardson et al., "Virtual Network Computing". IEEE Internet Computing 1998
[8] Microsoft, "Windows Remote Desktop Protocol (RDP)". http://msdn2.microsoft.com/en-us/library/aa383015.aspx
[9] Citrix Independent Computing Architecture. http://www.citrix.com
[10] L. Deboosere et al., "Thin client computing solutions in low- and high-motion scenarios". In ICNS 07: Proceedings of the 3rd International Conference on Networking and Services. Washington DC, USA.
[11] P. Simoens et al.,"Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices". Proc. of Australasian Telecommunication Networks and Applications Conference (ATNAC) 2008, Adelaide, Australia.
[12] VideoLan – VLC Media Player, http://www.videolan.org/vlc/