30'6.
NETW1033

ICT-MobileSummit 2009 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2009
ISBN: 978-1-905824-12-0

Demanding Applications on Resource Constrained Mobile Devices through Thin Client Computing

Bert VANKEIRSBILCK, Pieter SIMOENS, Lien DEBOOSERE,
Filip DE TURCK, Bart DHOEDT, Piet DEMEESTER
Department of Information Technology (INTEC), Ghent University - IBBT
Gaston Crommenlaan 8 bus 201, 9050 Ghent, Belgium
Email: bert.vankeirsbilck@intec.ugent.be

Abstract: Due to battery power limitations and form factor consideration, mobile devices are typically resource constrained. This results in sub-optimal user experience for execution of demanding applications. The thin client computing paradigm shifts the applications to a distant server, relieving the mobile device from the heavy computations. However, existing thin client protocols do not handle multimedia content well and are not designed for wireless networks with varying bandwidth. We propose a hybrid approach that switches to streaming when highly dynamic screen updates are detected. For the case of static screen updates, a caching mechanism is presented that mitigates bandwidth usage, to limit the energy drained by the wireless network interface.

Keywords: Mobile Device, Thin Client Computing, Caching, Quality of Experience

1. Introduction

Because of form factor constraints, required portability and battery lifetime, mobile devices have limited resources. Despite the rapid development of smaller and more efficient hardware, software requirements on CPU, memory and disk space cause significant hardware needs and battery drains. The idea presented in this paper is to extend the mature thin client technology for use on mobile devices, where the heavy computations are offloaded to a distant server and the applications need no alteration. User input is sent over a network towards a server that executes the application, and the generated screen updates are returned to the user's device. The functionality of the mobile device is reduced to the presentation of graphical output, the capture of user events such as key strokes and pointer device movements, and transmitting data over the network.

A key challenge in a thin client system is to limit the delay between a user event, e.g. a keystroke, and the corresponding display update. Users are accustomed to the crisp graphical

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 nr 216946. Bert Vankeirsbilck and Lien Deboosere are funded by a Ph.D grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). Pieter Simoens is funded by a Ph.D grant of the Fund for Scientific Research, Flanders (FWO-V).

Copyright © The authors

interface of locally running applications and expect the thin client computing infrastructure to offer the same functionality [1]. Current remote display protocols such as Virtual Network Computing's Remote Frame Buffer protocol (VNC-RFB) [2] and Microsoft's Remote Desktop Protocol (RDP) [3] are optimized for the rather static display of office applications, such as text editor and spreadsheet applications, over a stable, fixed, high bandwidth network.

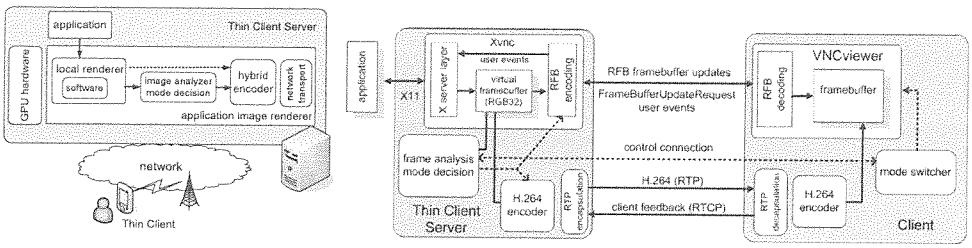
In this paper an adaptive protocol is presented that bases it's adaptations on the content of the screen updates in order to optimize bandwidth usage for the application at hand. If there is a lot of motion on screen, the protocol uses H.264 streaming [4] to ensure sufficient Quality of Experience (QoE) consuming less bandwidth than a classic thin client protocol. In the other case, when the frequency of graphical updates is low and the content is rather static, an optimization through caching is presented as to lower bandwidth usage of the classic protocol. The switching decision is taken on the rendered-pixel level without user intervention, so it is essentially application and user independent. After an overview of related work, Section 3 explains the architecture of the hybrid protocol. Section 4 elaborates on the dynamic case. Through experimental results, it is explained and proven that benefit can be taken from the presented hybrid protocol. In section 5, an optimization for the static case is presented, and experimental results show a significant gain can be reached by using a limited amount of reference frames.

2. Related Work

Although thin client technology exists for more than two decades, using the paradigm straight away over wireless communication channels is problematic. In [5] the authors evaluate the performance of existing protocols, and conclude that they are optimized for fixed LAN and WAN environments. Over a wireless infrastructure there is room for improvement, as can be deduced from [6]. The main issues to be addressed for thin client computing protocols to work over wireless networks are delay, bandwidth and in certain cases the QoE.

In [7] the authors identify the presence of data spikes and their importance for momentary network load. Peaks in bandwidth demand typically cause high delay (because of buffering (and eventual router queue overflow), packet loss and retransmissions), leading to hampering user experience. The solution presented in [7] is to cache fixed-length byte strings selected from data packets representing a part of a compressed graphical update. [8] shows that up to 23.3% network traffic reduction can be achieved by an intelligent extension scheme for the compression history buffer. Both papers focus on extending and tuning a compression scheme to the specific thin client traffic. In FreeNX [9], graphical update commands (rather than pixel data) are cached. Two popular thin client protocols are Microsoft Remote Desktop Protocol (RDP) [3] and Citrix Independent Computing Architecture (ICA) [10]. In contrast to our full frame caching and redundancy reduction attempt, both solutions cache smaller parts of the screen.

Copyright © The authors



(a) architecture (b) implementation

Figure 1: Architecture and implementation detail of the hybrid thin client protocol.

3. Architecture

In Figure 1(a), an overview is presented of the architecture of a thin client server. A mobile device is wirelessly connected to the thin client server where the user's applications are executed. To render its images, an application issues graphical calls to an application image rendering component. This component offers an unmodified interface to the application for its graphical calls. The application is therefore unaware that it is running in a thin client environment and not locally on the device, since no modifications are required to run existing applications in this thin client architecture.

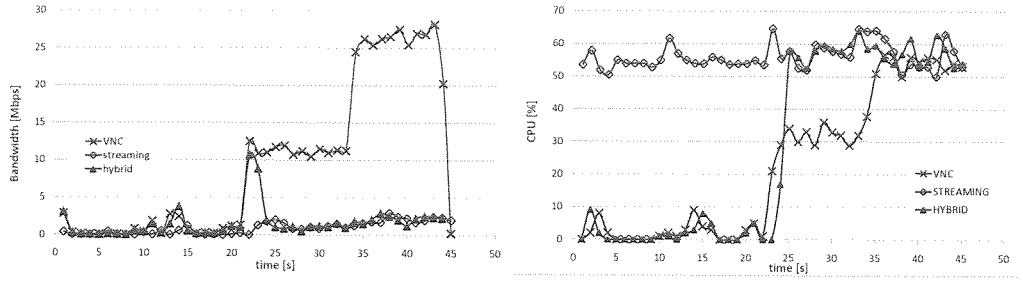
The application image rendering process is composed of the following steps. The graphical calls issued by the applications are received by the local renderer, which renders the images through software rendering and/or by using dedicated Graphical Processing Unit (GPU) hardware if present on the thin client server. The rendered images are analyzed and a decision is taken on the mode in which they will be relayed to the client. A hybrid encoder generates from the image stream either a remote display protocol message stream or a video stream, before they are sent to the client. Thus, the decision is not steered by user intervention, but by the graphical content that is generated on screen.

4. Dynamic Content: Streaming

4.1 — Concept and Architecture

As explained above, a hybrid protocol has been designed that switches seamlessly between encoding the screen update through a classic thin client protocol or through streaming. Figure 1(b) presents the detailed architecture of this hybrid protocol. The decision of which encoding to use, is based on real-time analysis of the raw pixel data of the screen, which can be extracted from the virtual framebuffer. When there is a lot of difference between subsequent frames, this decision module selects streaming as the best option to encode the frame updates. This module incorporates hysteresis so that needless switching can be avoided.

We implemented the hybrid protocol by extending a VNC server and client. This protocol divides the content of the frame buffer into rectangles. Several 'flavors' of VNC exist, differing in the applied coding on these rectangles. Previous measurements have designated



(a) bandwidth (b) client CPU

Figure 2: Resource requirements for the hybrid protocol.

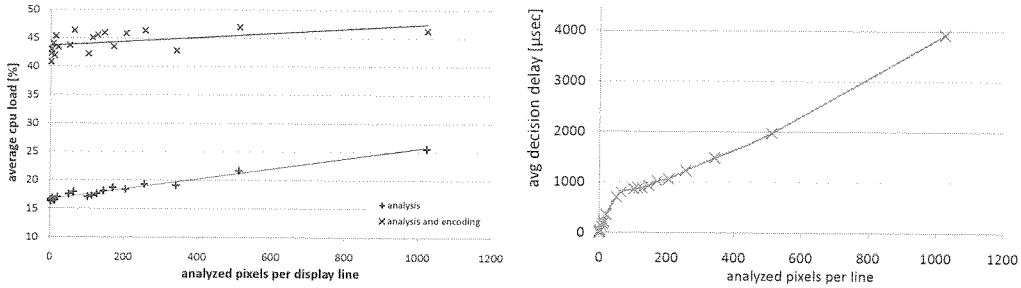
the TurboVNC coding as the most bandwidth efficient [11]. For the streaming mode, the well-known H.264 codec [4] was selected, transported over the Real-time Transport Protocol (RTP) [12]. Figure 1(b) shows the implementation of the application image renderer and the client viewer.

4.2 — Experimental Validation

Figure 2 depicts the resource requirements for the hybrid protocol in terms of bandwidth and client CPU. The client was an Intel 1.73 GHz machine with 1 GB RAM, the server was an Intel QuadCore 2.73 GHz server with 2 GB RAM. The measurements were taken for a sequence of about 45 s, in which a user respectively opens a text editor, enters text and includes a figure. At time $t = 23\ s$, the VLC video player is opened to watch a video, but not in full-screen. Around $t = 35\ s$, the video window is maximized to full-screen (1024x768). This sequence was recorded with the *cnee* [13] tool and replayed in 3 different modes: only VNC, only streaming and the hybrid approach. By comparing the hybrid line to the VNC and streaming curves on the following graphs, one can clearly see which mode is activated.

Figure 2(a) shows the advantage of the streaming approach for video or other high-motion content. While VNC requires a bandwidth of about 28 Mbps, the streaming approach only requires around 2 Mbps to deliver the images to the client. Since VNC and streaming take the same amount of bandwidth for low-motion sequences, one could conclude that even in this case, the streaming approach could be used. For mobile thin clients, the client side CPU load is important to consider. Figure 2(b) indicates that decoding RFB (VNC) messages is much less resource intensive compared to video decoding. This advocates the use of VNC for applications with statical graphical content, offering the same user experience as video streaming but requiring much less CPU cycles.

At the server side, it is important to assess the CPU load and delay incurred by the analysis of each frame. The results for both CPU and analysis delay, averaged over 20 iterations, are shown in Figure 3. In order to make the pixel analysis scalable, a regular grid of sample pixels was chosen. The size of the grid was varied, obviously influencing the server side CPU load and overall encoding delay, such that the total number of analyzed pixels is between 0.1%



(a) server CPU (b) analysis delay

Figure 3: Server CPU load and analysis delay for the hybrid protocol implementation, averaged over 20 iterations.

and 50% of the total number of pixels in the display. As can be deduced from Figure 3(a), the CPU load induced by the hybrid server averages around 45%. The indicated CPU load is the load of a single core on our Intel 2.73 GHz Quad Core, which means that our server machine could handle about 8 high resolution clients. However, the depicted CPU load is an average over the trace, comprising both text editing, with minor display changes, and watching a video, with more frequent display changes. During the video streaming part of the trace, the CPU load (on a single core) induced by the hybrid server, can amount up to 100%, while during the text editing part, the CPU load drops to less than 5%. The figure also indicates that a linear correlation can be distinguished between the number of pixels analyzed per line and the CPU load, ranging between 15% and 25%.

It is important to minimize the additional processing delay at the server due to the image analysis and encoding. If we want to generate a framerate of at least 25 Hz at the server, the total time to analyze and encode a frame should be kept much below 40 ms. In our implementation, the *libavcodec* [14] library was used for the video encoding. Measurements have shown that the encoding of a single frame takes about 30 - 35 ms. To keep up with the frame rate of 25 fps, the time required for the frame analysis and mode decision must be kept below 5 ms. Figure 3(b) shows the additional delay incurred by the pixel analysis, for different numbers of sample pixels.

5. Static Content: Graphical Update Caching

5.1 — Concept and Architecture

In the previous section, experimental results indicate that for static content, a classic thin client protocol is a better encoding choice than streaming. The current thin client protocols are performing adequately in a stable, high bandwidth network environment. However, mobile device are battery powered, and enjoy all benefit of using less bandwidth as this lowers the use of the wireless network interface and thus mitigates the related power consumption.

From analysis of typical thin client traffic, we have concluded that momentary bandwidth

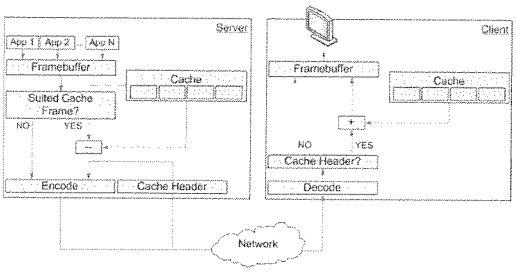


Figure 4: Caching architecture.

peaks often resemble updates caused by switching between the running applications. In this section we propose the use of a cache that stores the start-up screens of the user's applications and the background screen of the operating system. This cache is constituted of frames that are selected before the thin client session starts and does not change during this session. The selection is thus based on the expected graphical content that will be generated by the user.

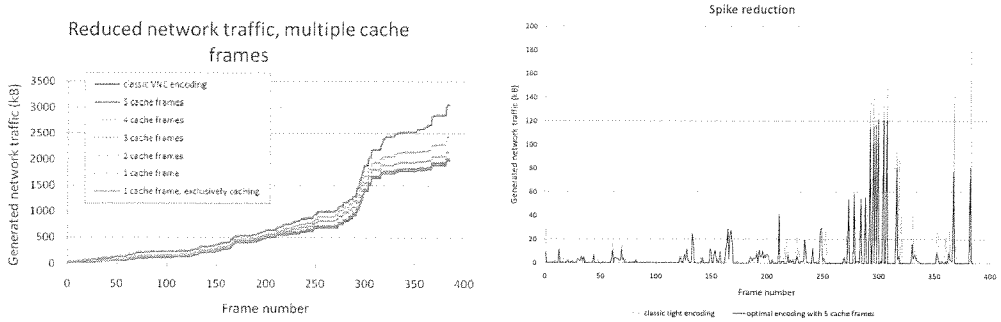
The architecture for integrating a static cache in a thin client computing system is shown in Figure 4. As with the decision module of the hybrid protocol, pixel data from the framebuffer is analyzed in order to choose the best cache frame. If there is a well matching cache frame available, it is subtracted from the graphical update and the difference is encoded using a classic encoding scheme and sent over to the client. In addition, a cache header containing necessary parameters such as the used cache frame has to be sent over to the client. At the client side, the received data is decoded, and depending on the presence of a cache header, the indicated cache frame is added to the decoded frame and delivered to the client framebuffer which eventually is presented on the screen of the client device.

5.2 — Experimental Validation

The proposed architecture for the static cache has been based on VNC as well. The measurement trace consisted of showing the desktop background; opening a command shell window (with black background) and typing a number of commands; entering a text in Open Office Writer, which was then minimized; performing a Google search through a browser; visiting a local newspaper's homepage and scrolling down; closing the browser and the shell window; maximizing the office program and closing it, and ending the thin client session from the desktop background.

A static cache containing five cache frames was used and was kept unaltered during the session. The cache frames were chosen off-line to be the startup screens of the used applications: the desktop background, an empty Open Office Writer document, a Konsole command shell window, a web browser with Google search page loaded and a web browser with the homepage of the local newspaper loaded.

The measurements were performed on 2.11 GHz AMD Athlon 64 X2 Dual Core machines, both for server and viewer, running Kubuntu 8.04.



(a) Cumulative network traffic

(b) Momentary network traffic

Figure 5: Influence of cache frames on average and momentary bandwidth usage.

Figure 5(a) shows a cumulative bandwidth usage of exactly the same trace encoded with a number of cache frames varying between zero and five. The figure shows gradual increase in bandwidth gain caused by addition of an extra cache frame. The use of one cache frame, the desktop background, already yields 20.56% bandwidth reduction. Adding an extra cache frame, the browser with the homepage of the local newspaper, brings the total bandwidth gain to 29.94%. With five cache frames we achieved a bandwidth consumption decrease of 34.40% over classic VNC encoding all updates.

Figure 5(b) presents the momentary network traffic generated by the classic VNC encoding and the optimal encoding using a cache with five frames. It shows that for momentary network traffic generation, the frames were coded on average 35.17% more efficient than through classic VNC encoding. The maximum spike reduction in the trace amounted to 99.81% of the classic VNC encoded update. The highest spike that occurs with the classic VNC encoding is 181.132 kB. Using five cache frames reduced the spike maximum to 121.175 kB, a reduction of 33.10%.

6. Conclusion

In this article a hybrid remote display protocol is presented that switches between two modes of relaying screen updates to the client, depending on the content and frequency of these updates. For complex graphic updates at high rates, streaming the content is the best option. For rather static graphics, it is more efficient to use a remote display protocol such as VNC-RFB. For the latter, caching has been proposed to lower the needed bandwidth. The decision to switch between the two modes is taken without user intervention, and is based on the graphical content (in pixel format) on the screen. Multimedia applications typically generate dynamic graphics which will trigger streaming mode, while office-type applications generate static graphics which will trigger VNC-mode. However, the analysis algorithm is application-independent, so that rapidly scrolling through a large text document could as well imply a switch to streaming mode and pausing a video in a multimedia application could imply

switching to VNC-mode. Future work includes designing a protocol that adapts even more to the environment it is operating in (the network conditions, client battery status, latency, etc.).

References

[1] N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *IEEE Computer*, vol. vol. 39, no. 3, pp. 46–52, 2006.

[2] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual network computing," *IEEE Internet Computing 1998*.

[3] "Windows Remote Desktop Protocol (RDP)." http://msdn2.microsoft.com/en-us/library/aa383015.aspx.

[4] "ITU-T: H.264, Advanced video coding for generic audiovisual services," 2005.

[5] A. Lai and J. Nieh, "On the performance of thin client computing," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, pp. 175–209, May 2006.

[6] J. Nieh, S. Yang, and N. Novik, "A comparison of thin-client computing architectures," tech. rep., CUCS-022-00, Department of Computer Science, Columbia University, November 2000.

[7] S. Yang and T. T. Tiow, "Improving interactive experience of thin client computing by reducing data spikes," *6th IEEE/ACIS International Conference on Computer and Information Science*, pp. 627–632, July 2007.

[8] S. Yang and T. T. Tiow, "Long distance redundancy reduction in thin client computing," *6th IEEE/ACIS International Conference on Computer and Information Science*, pp. 961–966, July 2007.

[9] "NoMachine FreeNX." http://www.nomachine.com.

[10] "Citrix Independent Computing Architecture (ICA)." www.citrix.com.

[11] L. Deboosere, J. De Wachter, P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Thin client computing solutions in low- and high-motion scenarios," in *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, (Washington, DC, USA), p. 38, IEEE Computer Society, 2007.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications." RFC 3550 (Standard).

[13] "Gnu xnee." http://www.gnu.org/software/xnee.

[14] "Ffmpeg." http://ffmpeg.mplayerhq.hu.

Copyright © The authors

ICT-MobileSummit 2009

10 - 12 June, Santander, Spain

Preface

In the context of convergence, the 18th ICT-Mobile Summit addresses all the challenges of building the Future Internet, which will be based on mobile, wireless and fixed broadband communications infrastructures.

Supported by the European Commission and eMobility and Technically Co-sponsored by IEEE, the ICT-MobileSummit's reputation is based on high quality paper and poster sessions that showcase original results in all areas of wireless communications systems and networks, including Mobile and Fixed, Terrestrial and Satellite. All papers are double blind peer reviewed by at least two Members of the Technical Programme Committee.

The ICT-MobileSummit 2009 Conference Proceedings gathers together a comprehensive collection of over 150 paper and poster contributions from Europe, North America and Asia sharing insight, cutting edge research and good practice.

Reflecting the breadth and depth of research undertaken by the contributors, the contents are broken down into the themes: Radio Access and Spectrum, Converged and Optical Networks, and Future Internet Technologies. Papers within each thematic area are grouped as Issues, Applications, Case Studies and Poster Papers, reflecting their primary focus.

We would like to acknowledge the valuable contribution of the Technical Programme Committee who provided authors with actionable feedback in finalising their papers for publication, and the encouragement and support of the European Commission and eMobility.

Paul Cunningham

Supported by

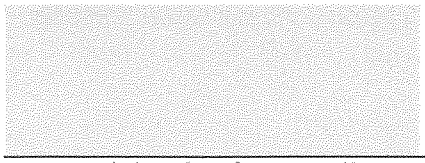
European Commission Information Society and Media

eMobility (Connect to the network community)

UC UNIVERSIDAD DE CANTABRIA

Technical Co-Sponsor

IEEE Celebrating 125 Years of Engineering the Future



Miriam Cunningham

Powered by ConferenceManager


ICT-MobileSummit 2009
Conference Proceedings
ISBN 978-1-905824-12-0
To view Conference Proceedings
open 'toc.html'
10 - 12 June 2009
Santander, Spain
www.ICT-MobileSummit.eu