DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

Improving the RDP based applications by using HTML5 content representation

Rama Rao Ganji[1], Mihai Mitrea[1,2], Dancho Panovski[1], Bojan Joveski[2]; [1]Institut Mines-Telecom ; Telecom-SudParis ; UMR 8145 – MAP5 and [2]Ustartapp (France)

Abstract

Despite the large variety of "off the shelf" solutions and academic research studies, application virtualization for cloud distribution is still an open research topic, with significant issues to be solved, ranging from bridging the gap between users expectation of simple, intuitive and user-friendly access from any type of terminal and the fragmented landscape of SaaS offer, with peculiarities related to the hardware/software configurations and the optimization of the technical resources consumption.

Our study investigates the possibility of using HTML5 a virtualization tool for RDP-based applications. Architectural modules related to the RDP content interception, conversion, adaptation, remote rendering and interaction are specified, designed and implemented. This architecture is validated under the framework of the MEDUSA European project, in partnership with medical institutions. The testbed considers a server and 5 mobile users, with heterogeneous devices (tablets, smartphones, laptops) running under iOS, Android and Windows operating systems. The objective/subjective evaluations demonstrated that: (1) the user experience is not reduced by the virtualization, (2) the network consumption is reduced by a factor of 1.8 with respect to state-of-the-art solutions.

Introduction and state-of-the-art

The mobile devices proliferation and the virtualization of the applications in the cloud raise the need of a solution that will answer the user exigencies. Under this framework, defining a virtualization mechanism suitable for any type of mobile thin client remains a challenging research topic: ensuring a high performance compression algorithm for heterogeneous content and affording versatile, user-friendly and real time interaction are issues to be jointly dealt with. The underlying technical deadlocks are mainly connected to the network (between the cloud and the terminal) and to the terminal capabilities.

However, the nowadays state-of-the-art is very broad and reach, see Table 1.

Joveski et al. [2][3] addresses the existence of large variety of "off the shelf" solutions, but almost all of the solutions are based on RDP (Remote Desktop Protocol)[5] and RFB (Remote Frame Buffer).

RDP is a proprietary protocol developed by Microsoft, which provides a visual/audio description (graphics, images, video, music) generated by the applications that are running on a remote Windows operating system. The protocol specification is public, thus encouraging a development of new type of applications based on RDP. Typically RDP is used to remotely access the applications, executed on a Windows server, by using a Windows RDP client.

RFB is remote access protocol to graphical user interfaces that works on the framebuffer level. The framebuffer provides only the pixels information, thus making the protocol on the one hand universal (all the types of operating systems) but on the other hand very limited (ignores the graphical primitives available). Virtual Network Computing (VNC) and its derivatives mainly use RFB.

As VNC protocol doesn't support the 3D applications, Deboosere et al. [10] proposed solution for thin clients virtualization based on low-motion and high-motion scenarios in Linux/Unix applications. They used VirtualGL [13] open-source project which enables 3D rendering in a thin client. VirtualGL helps to send OpenGL commands to GPU at the server side and reads back the rendered images. The other graphical commands (X commands) are send to X server. VirtualGL operates in two modes: Raw and Direct Mode. In VirtualGL Raw Mode, the virtual X Server (X Proxy) is used in order to receive all the graphical content and to send them to thin clients. The thin client user interactions like mouse/keyboard events are send to X Proxy (TurboVNC). In VirtualGL Direct Mode, the thin client has two libraries: 1) X Server who takes care of all the standard Xlib functionality and 2) a VirtualGL client who decompresses the image stream sent by VirtualGL Direct Mode. The results are illustrated for two scenarios. First, in the low-motion case, both text editing (typing, scrolling, inserting….) in Open Office 2 Writer and browsing (a sequence of website visited in Mozilla Firefox 2.0) are considered. Secondly, in high-motion scenario, a user is alternatively watching a video with VLC / Windows Media Player 10 or the content generated by a 3D-game (Unreal Tournament 2004). The results are benchmarked according to the CPU usage at the client side as well as to the network bandwidth consumption. All these results are compared to various other state-of-the-art solutions like RDP, CitrixICA, VirtualGL Direct [13], FreeNX ADSL, VNC Turbo, VNC Tight, VNC standard, VNC Hextile. It is concluded that not all the protocols are able to support high-motion visual content. Moreover, in the case of 3D games, the protocols which are able to run smoothly (VirtualGL Direct mode and VirtualGL Raw mode) require a very high bandwidth, thus becoming prohibitive in mobile environments. In order to overcome this limitation, Weidong et al. [11] proposed a solution based on Virtual OpenGL driver and converting the images into MPEG-4 AVC (H.264) video stream to reduce the network bandwidth consumption.

Weidong et al. [11] proposed SHARC (**S**calable 3D Grap**h**ics Virtual **A**ppliance Deliver**y** in **C**loud) to address the virtualization of 3D applications like video games in cloud. As 3D applications require very resource intensive computation and graphics, SHARC uses virtual OpenGL driver [13]. The Graphics Rendering Server receives the 3D graphics from multiples applications, and generate custom JPEG stream that is sent to media streaming server. Upon receiving this stream, the Media Streaming Server converts it to MPEG-4 AVC video stream and sends it to the clients.

Rodríguez-Silva et al. [12] proposed (VIMAIN) the virtualization of the applications based on QEMU-KVM open

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

source hyper visor. The virtualization is based on the VNC server and WebM [14] video streaming. When the applications are generating low-motion content, the graphical output of the application is send by using the VNC server. When the application generates high-motion graphical content, the graphics are encoded as video in WebM format, and then sent to the clients. The virtualization is independent of the application operating system.

The experimental results are conducted on both subjective and objective basis. The subjective evaluation considers 20 users who are inquired about the quality of experience with 5-slide PowerPoint slideshow. It was concluded that the user experience is better when WebM video streaming is used for high-motion scenarios. The objective evaluation is represented by a benchmarking against RDP, VNC-raw, VNC-zlib, VNC-tight, and OnLive. The minimal bandwidth is required by VIMAIN; however, this is achieved at the expense of increasing the CPU (mainly for video encoding).

Table 1. Synopsis of state-of-the-Art solutions

	Method		Specification	Remarks
Off- the-shelf solutions	VNC	application	Desktop applications	
		content	Images	Based on RFB protocol
		interaction	Operating system	Keyboard / Mouse
	RDP client	application	Desktop applications	Windows applications
		content	Graphics, 2D, 3D, audio, video	Additional setup at client side
		interaction	Operating system	Keyboard / Mouse
Research study	Deboosere et al.	application	3D	Linux applications
		content	Solution based on VirtualGL	3D & X11
		interaction	/	Not part of the study
	Rodríguez Silva et al.	application	Desktop applications	Any type, targeted 2D
		content	RFB images + WEBM video	
		interaction	/	Not part of the study
	Weidong et al.	application	Gaming	3D games
		content	converts the images sent by VirtualGL into H.264 video stream	Video
		interaction	/	Not part of the study
	Joveski et al.	application	Desktop applications	Initially Linux(2D) and later extend it to windows applications
		content	Graphics, 2D, audio, video	
		interaction	yes	

In all the above cases, regardless of its original type, the heterogeneous graphical content (text, image, graphics, video, 3D, …) generated by the application is converted into sequences of images (eventually a mixture of images and graphics), which are subsequently interactively displayed on the terminal.

In order to ensure a true multimedia experience, Joveski et al [15][3] advanced an architecture which intercepts the graphical content generated by the application (text, image, video, 2D/3D graphics) and converts it into an MPEG-4 BiFS multimedia scene. This scene is subsequently adapted, compressed and stream towards the mobile client, where it is rendered inside a MPEG-4 BiFS player. Initially they considered only Linux applications and they further extend it to include [2] Windows applications by accessing them via Remote Desktop Protocol (RDP).

In our previous study [1] we provided a way to virtualize the Linux applications by intercepting X11 commands and converting them to HTML5 <canvas> elements. The converted graphics are compressed and streamed towards the HTML5 supported clients.

Advanced architecture:

The present paper reconsiders and extends the client-server architecture presented in [1][2][16]. The following components are developed: *Application Execution, Content analysis, HTML5 description, Pruning* and *Compression & streaming,* see Figure 1.

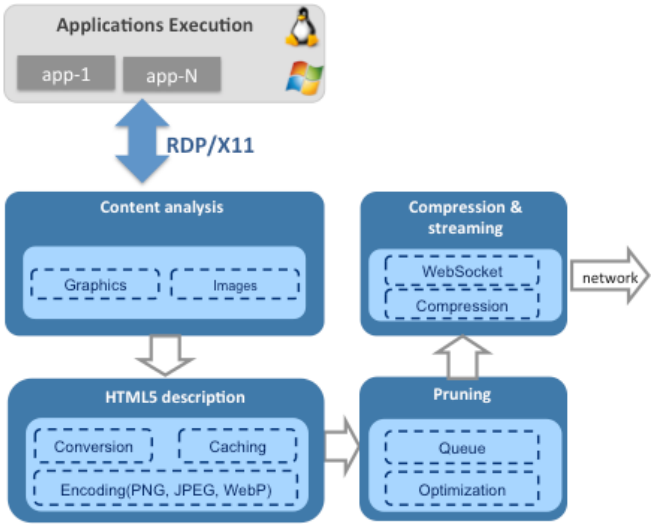

Figure 1. Improved architecture for HTML5 content representation.

Application Execution:

The *Application Execution* is standalone module where the applications are executed and their graphical output is captured and forwarded to *Content analysis* component via RDP [5].

Content analysis:

The *Content analysis* component interprets the RDP content and identifies the graphical primitives like rectangle, lines, glyphs and images… All these graphical primitives and images are sent to HTML5 description component in order to convert them into HTML5 <canvas> elements graphics, as exemplified in Table 2.

HTML5 description:

HTML5 description contains three modules *Conversion, Encoding and Caching*. The graphical primitives received from *Content analysis* are converted into HTML5 <canvas> element graphics by describing them in JavaScript. Moreover the images are encoded using the appropriate compression algorithms like

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

PNG, JPEG, or WebP, according the browsers capabilities (for example WebP is only supported by Chrome).

After the content is converted into HTML5 and before sending to the Pruning module, it is cached. This caching component allows us to buffer the content for several milliseconds for further analysis by the Pruning module.

Table 2. Conversion of RDP graphics into HTML5 <canvas>

RDP content	HTML5 <canvas> conversion
LineTo: to draw line in RDP *Required fields*: XStart, YStart, XEnd, YEnd, BackgroundColor, PenStyle, PenWidth, PenColor	*dL*: draw the line on canvas *Required fields*: xstart,ystart,xend,yend, background_color, line_style, line_width, line_color
OpaqueRect: Drawing opaque rectangle. *Required fields*: LeftRect, TopRect, Width, Height, Red, Green, Blue.	*dR*: *Required fields*: xstart, ystart, width, height, RGB
ScrBlt: copy a rectangle area from source position to destination position. *Required fields*: LeftRect, TopRect, Width, Height, Rop, XSrc, YSrc.	copyArea: fucnction copy area from source to destination *Required fields*: srcX, srcY, destX,destY, width, height, rop
MemBlt: render a cached bitmap stored bitmap cache or offscreen bitmap cache *Required fields*: cacheId, leftRect, TopRect, Width, Height, SrcX, SrcY, Rop, cacheIndex.	*drawbitmap*: *Required fields*: src_id, dst_id, srcx,srcy,width, height, dstx,dsty,rop

Pruning:

The main functionality of the Pruning module is to optimize the HTML5 content by analyzing the images and rectangles that are generated. Usually, when images and rectangles are generated very fast on a small piece of area, the pruning algorithm makes a decision of the number of images and/or rectangles to be sent. This decision is based on the network and device capabilities, and the number of updates.

For example, in the Figure 2, the seven updates from 1 to 7 can be considered as images/rectangles. The Pruning mechanism analyses each update (position size) and checks whether it's possible to remove some updates based on. In the example, the result is remove the update 1.

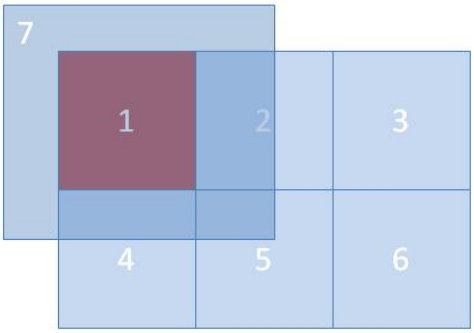

Figure 2. Pruning mechanism example

After the optimization is performed the content is queued and ready to be compressed and streamed.

Compression & streaming:

The compression component applies the browsers and websocket supported compression, permessage-deflate [1]. The WebSockets server establishes a bidirectional connection with the clients' browser and is kept open during its usage. The websocket also is in charge of receiving the user interactions like mouse move/click, key click, touch etc…

Experimental Setup

The experiments consider 5 users, each of which using three different Windows applications: Internet Explorer version 9, MS Word version 2010, DICOM image viewer [18].

The usage of each application respects a pre-established scenario, including heterogeneous interaction modes and generating various types of visual content, as explained below.

For text editing, each user is typing for 5 minutes the beginning of the Plato's Republica. According to their typing speed, they generate between 550 and 1100 characters.

For the Internet browsing, each user performs a 9 step scenario: (1) load Google page, (2) type "Wikipedia mobile", hit enter and wait for the page to be loaded, (3) click the Wikipedia mobile link and wait for the Wikipedia page to be loaded, (4) type "chocolate" in the search area, hit enter and wait for the searched result page to be displayed, (5) click the link "bitter" and wait for the new page to load, (6) click the "Bookmark" menu item, select the google.news link, and wait for the page to load, (7) click the home icon, and wait for the www.debian.org home page to load, (8) three times scroll down, (9) click the exit icon.

For the Dicom viewer, each user performed the following six steps: (1) start Dicom viewer application (2) open one Dicom image (3) click menu Image →Color Map → Select Color Map → smart (4) click on Image information icon (5) click the Line icon on side bar, and measure the distance between two points, and delete the line (6) Close the application.

The Windows applications are running inside a windows (version 7) virtual machine.

The client terminal is of Samsung Galaxy Tab 10.1 (model GT-P7510) Running OS Android 4.0.4. The client terminals are installed with the Chrome browser, Firefox browser and bVNC application (for the evaluation purposes).

The client terminal is connected to the server through the wireless network.

In-order to benchmark our HTML5 based solution we consider: 1) network bandwidth consumption, 2) CPU consumption and 3) image quality.

On the one hand, when evaluating the network bandwidth, the HTML5 based solution is considered two cases: 1) compressed and 2) uncompressed. On the other hand, when evaluating the CPU consumption, the HTML5 based solution is considered two cases: 1) accessing from Chrome and 2) accessing from Firefox.

All the results are benchmarked against the VNC Raw, VNC Hextile and VNC Tight [17] based solutions.

The measurements are obtained in the following setup:

- The applications are running on windows 7 virtual machine
  - RAM: 3GB
  - Process: 2
  - Applications: IE version 9, MS Word 2010.
- Imaging Client is running on Ubuntu 14.04 server.
  - RAM: 2GB RAM
  - Process: 2

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

- The client terminals consists of the following
  - Samsung Galaxy Tab 10.1 (model GT-P7510) Running OS Android 4.0.4
    - Memory:1GB
    - Hard disk: 16GB
    - Process: Nvidia Tegra 2 dual core 1Ghz
    - Chrome Browser
    - Firefox Browser
    - bVNC
- Network
  - Belkin- N6000 DB Wireless N+ Router

Experimental Results

Network bandwidth measurements

For the text editing experiment, the values (in KBytes) of the bandwidth required by the corresponding cumulative down-link traffic, averaged over the 5 users, are plotted as a function of time (indexed in minutes) in Figure 2; the value "0" on the abscissa refers to the scene initialization.

Note that in this experiment, the number of scene updates varies with the scene updates generated by each user (i.e. with the number of letters they actually typed in each time interval).

The www browsing experiment is illustrated in Figure 3, where the values (in KBytes) of the cumulative network traffic, averaged over the 5 users, are plotted as a function of the 9 steps.

The DICOM image viewer experiments are illustrated are in Figure 4, where the values (in Kbytes) of cumulative traffic, averaged over the 5 users, are plotted as function of 6 steps.

The HTML5-Compression performs better than the VNC HEXTILE (by factor of 1.33) and VNC RAW (by factor of 7.98) when 5 users are typing. However when compared to VNC Tight the network bandwidth required VNC Tight outperforms the Imaging client by factor of 1.14

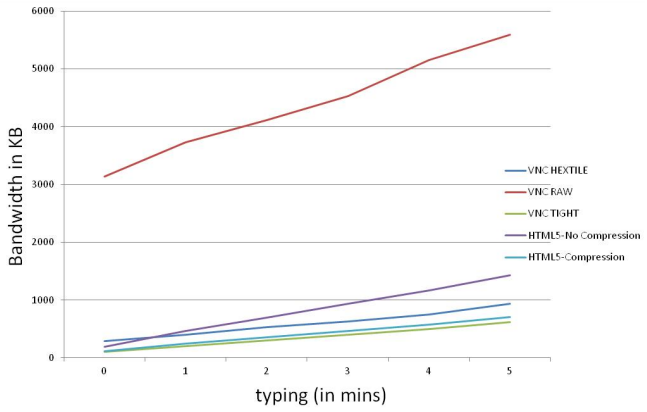

Figure 2. MS Word typing for 5 Mins in KB (kilo bytes)

For Internet Explorer, HTML5-Compression performs better than VNC Hextile by a factor of 2.2 and VNC Raw by a factor of 7.97. But VNC Tight outperforms the HTML5-Compression by a factor of 2.45.

For Dicom viewer, HTML5-Compression performs better than VNC HEXTILE (by factor of 1.10) and VNC RAW (by factor of 5.4). However, when we compared the VNC Tight with HTML5-Compression, VNC Tight outperforms the HTML5-Compression by a factor of 3.4.

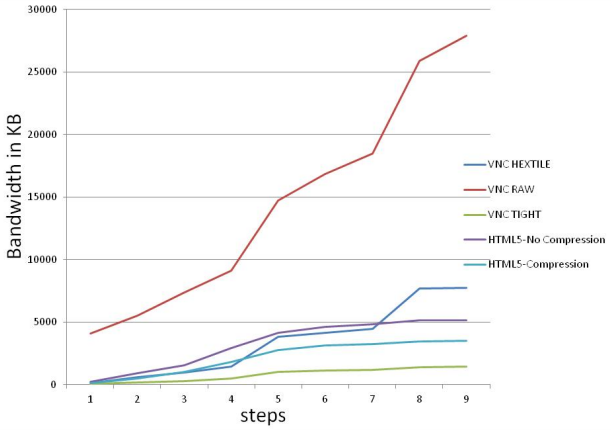

Figure 3. IE Browsing steps 1 to 9 and network consumption in KB (kilo bytes)

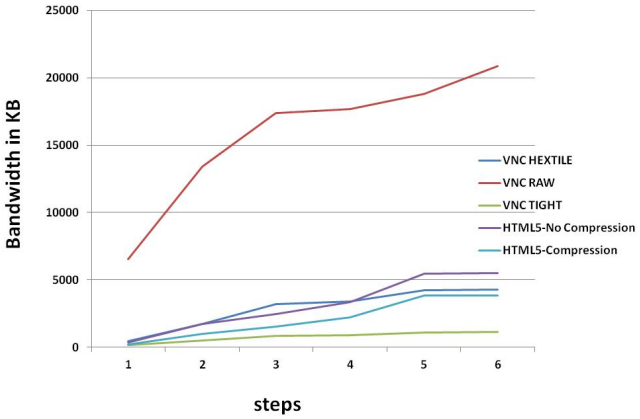

Figure 4. DICOM viewer steps 1 to 6 and network consumption in KB (kilo bytes)

CPU measurements

This section assesses the processor power (expressed in % from the total available CPU on the client device) needed to run the remote display in order to render all the received content.

In order to assess the CPU usage, we benchmarked VNC Hextile, VNC Tight, and HTML5 in two browsers, namely Chrome and Firefox. When executed in Android operating system, the Chrome browser automatically launches three processes, namely the chrome process, privileged process and sandboxed process. Consequently, the average and maximal CPU activity for Chrome browser, was measured by considering the average and maximal value over three process.

Both the average and the maximal CPU usage (over the 5 users) are evaluated as a function of time (or steps) and reported in Figure 5 to Figure 10, for MS Word text editing, Internet Explorer www browsing and DICOM Viewer, respectively.

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

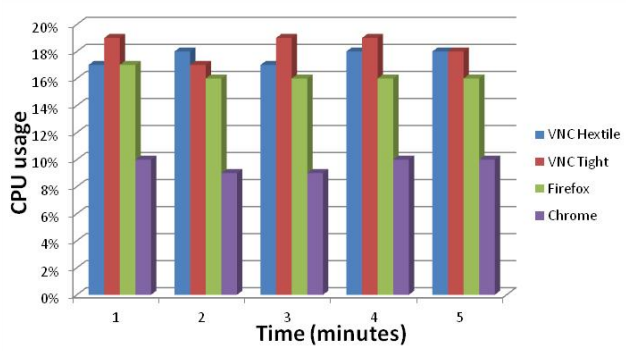

Figure 5. Average CPU peaks during MS Word typing after each min

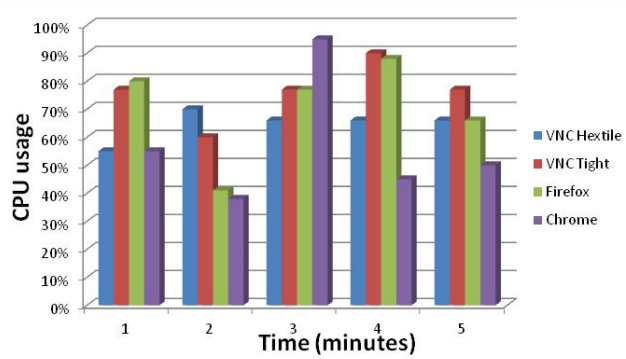

Figure 6. Average Maximum CPU peaks during MS Word typing after each min

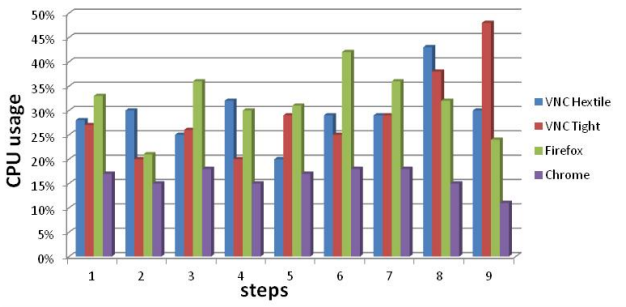

Figure 7. Average CPU peaks during IE browsing

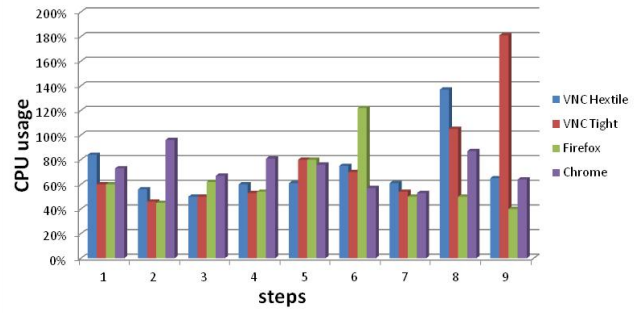

Figure 8. Max CPU peaks during IE browsing

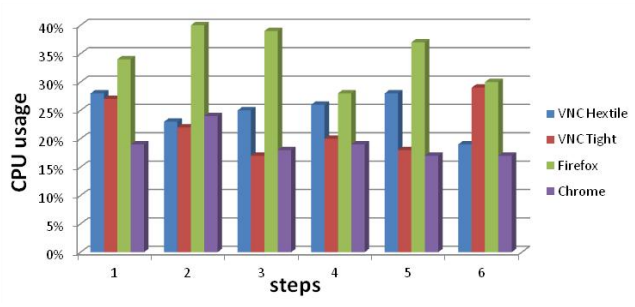

Figure 9. Average CPU peaks during DICOM viewer

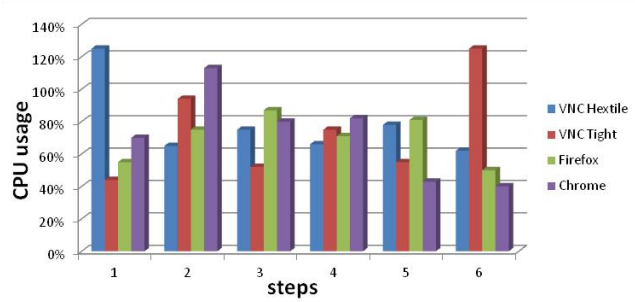

Figure 10. Max CPU peaks during DICOM viewer

The Maximum CPU usage for MS Word typing is Chrome by reaching a peak of 95%, for Internet Explorer is VNC Tight by reaching a peak of 180% (considering 2 CPU available for total of 200%), for DICOM viewer is VNC Hextile and VNC Tight by reaching a peak of 125%.

Image quality:

The process of conversion from RDP to HTML5 intrinsically introduces some differences between the original and the converted visual representations. The aim of this sub-section is to evaluate the artifacts induced by such a conversion.

Figure 11, Figure 12 and Figure 13 illustrate the quality of the converted content, represented on the HTML5 <canvas>, for the three above-mentioned experiments. No illustration has been done for VNC RAW, VNC HEXTILE and VNC Tight, as their visual content is kept unchanged during the transmission and displaying.

With a simple visual inspection Figure 11, Figure 12 and Figure 13, no visual disturbing artifacts can be identified: graphics, icons and text are spatiotemporally synchronized and colors/shadows are kept unchanged.

Consequently, we objectively evaluated the differences between the original RDP content and its HTML5 converted counterpart. The experiments considered six objective full-reference image quality measures of two types: (1) pixel difference based measures (PSNR - peak signal to noise ratio, and IF - image fidelity) and (2) correlation based measures (CQ - correlation quality, SC - structural content, NCC - normalized cross-correlation, and SSIM – structural similarity). The average values are presented with a 0.01 precision; the PSNR values are expressed in dB.

Note that as the VNC, VNC HEXTILE and achieves lossless image compression, their objective measures reach the ideal limits: PSNR $\rightarrow \infty$, IF = 1, CQ = SC = NCC = SSIM = 1.

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

The images are captured after each step and compared to the original images. For MS Word, there are more than 500 images, for Internet Explorer we captured 9 images after each step; similarly for DICOM viewer we captured 6 images for comparison.

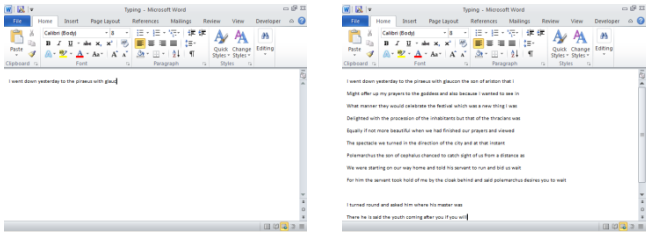

Figure 11. Screenshots of MS Word typing in in HTML5 <canvas>

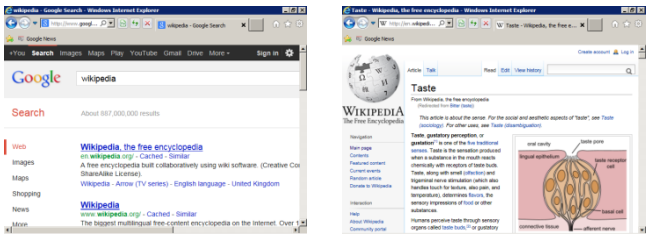

Figure 12. Screenshots of internet explorer in HTML5 <canvas>

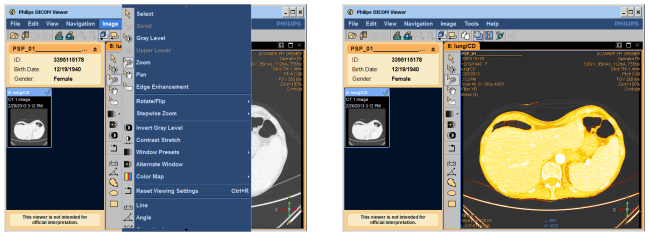

Figure 13. Screenshots of DICOM viewer in HTML5 <canvas>

Table 3: Image quality

App/Metrics	MSWord	IE	DICOM
PSNR	71.48	68.70	71.33
IF	0.999	0.999	0.999
SC	1.00	1.00	1.00
NCC	0.999	0.999	0.999
CQ	0.999	0.999	1.001
SSIM	0.999	0.999	0.999

Conclusion

The present paper provides the POC (proof-of-concepts) for the use of the HTML5 (Hyper Text Markup Language) as alternative virtualization tools for RDP-based applications (e.g. MS Windows applications).

From the methodological point of view, the main novelty consists in designing an architecture allowing the conversion of the RDP content into a HTML5 content representation and subsequently streaming this content by compressing to the clients where the HTML5 content is rendered.

The testbed considers a server and user devices. The experimental validation considers 5 users and three RDP applications (MS Word, Internet Explorer and DICOM viewer). The advanced solution is benchmarked against three state-of-the-art technologies (VNC Raw, VNC Hextile and VNC Tight). The visual quality is evaluated by six objective measures (e.g. PSNR>68dB, SSIM>0.99). The network traffic evaluation shows that: (i) for text editing, the HTML5-based solutions outperforms the VNC Hextile by a factor 1.33; however VNC Tight outperforms it by factor of 1.14 (ii) for Internet browsing, the HTML5 solutions outperform VNC Hextile by factors of 2.2 but VNC Tight outperforms it by factor of 2.4 (iii) for DICOM viewer, the HTML5 solutions outperform VNC Hextile by factors of 1.1 but VNC Tight outperforms it by factor of 3.4. The average CPU consumption for Chrome is lower than Firefox, VNC Hextile and VNC Tight for all the three applications. The maximal CPU usage for MS Word typing is Chrome, for Internet Explorer is VNC Tight, for DICOM viewer is VNC Hextile and VNC Tight.

Future work will be devoted to extending the HTML5 virtualization solution for Internet of Things devices and applications.

References

[1] R.R. Ganji, M. Mitrea, B. Joveski, F. Preteux, "HTML5 as an application virtualization tool," Consumer Electronics (ISCE), 2012 IEEE 16th International Symposium on , vol., no., pp.1,4, 4-6 June 2012, doi: 10.1109/ISCE.2012.6241695

[2] B. Joveski, M. Mitrea, R.R. Ganji, "MPEG-4 solutions for virtualizing RDP-based applications", Proc. SPIE 9030, Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014, 90300A (February 18, 2014); doi:10.1117/12.2042342.

[3] B. Joveski, M. Mitrea, P. Simoens,I. J. Marshall, F. Prêteux & B. Dhoedt, "Semantic multimedia remote display for mobile thin clients", Multimedia Systems,October 2013, Volume 19, Issue 5, pp 455-474, DOI 10.1007/s00530-013-0304-6

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

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

[6] Microsoft Office, https://products.office.com/

[7] Internet Explorer, http://microsoft.com/ie/

[8] WebSockets API, https://www.w3.org/TR/2011/WD-websockets-20110419/

[9] Websockets library, http://git.warmcat.com/cgi-bin/cgit/libwebsockets

[10] Deboosere, L.; De Wachter, J.; Simoens, P.; De Turck, F.; Dhoedt, B.; Demeester, P., "Thin Client Computing Solutions in Low- and High-Motion Scenarios," Networking and Services, 2007. ICNS. Third International Conference on , vol., no., pp.38,38, 19-25 June 2007

DOI: 10.2352/ISSN.2470-1173.2016.7MOBMU-293

[11] Weidong Shi; Yang Lu; Zhu Li; Engelsma, J., "Scalable Support for 3D Graphics Applications in Cloud," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , vol., no., pp.346,353, 5-10 July 2010; doi: 10.1109/CLOUD.2010.76

[12] Rodríguez-Silva, Daniel A., Jaime Loureiro-Acuña, Francisco J. González-Castaño, and Cristina López-Bravo. "Improving the virtualization of rich applications by combining VNC and streaming protocols at the hypervisor layer." Software: Practice and Experience (2015).

[13] VirtualGL, http://www.virtualgl.org/

[14] WebM, http://www.webmproject.org/

[15] Bojan Joveski “Semantic multimedia remote viewer for collaborative mobile thin clients” PhD theis., Ecole Nationale Supérieure des Mines de Paris, 2012

[16] Rama-Rao Ganji, Mihai Mitrea, Bojan Joveski, Afef Chammem, "Cross-standard user description in mobile, medical oriented virtual collaborative environments", in Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2015, Reiner Creutzburg; David Akopian, Editors, Proceedings of SPIE Vol. 9411 (SPIE, Bellingham, WA 2015), 94110G

[17] TightVNC, http://www.tightvnc.com/

[18] Philips Dicom viewer, http://clinical.netforum.healthcare.philips.com/global/Explore/Clinical-News/MRI/Philips-DICOM-Viewer-download-version-R30-SP3