Assistive Technology
The Official Journal of RESNA

ISSN: 1040-0435 (Print) 1949-3614 (Online) Journal homepage: http://www.tandfonline.com/loi/uaty20

Providing Assistive Technology Applications as a Service Through Cloud Computing

Davide Mulfari, Antonio Celesti, Massimo Villari & Antonio Puliafito

To cite this article: Davide Mulfari, Antonio Celesti, Massimo Villari & Antonio Puliafito (2015) Providing Assistive Technology Applications as a Service Through Cloud Computing, Assistive Technology, 27:1, 44-51, DOI: 10.1080/10400435.2014.963258

To link to this article: http://dx.doi.org/10.1080/10400435.2014.963258

Accepted author version posted online: 06 Oct 2014.
Published online: 06 Oct 2014.

Assistive Technology® (2015) **27**, 44–51
Copyright © 2015 RESNA
ISSN: 1040-0435 print / 1949-3614 online
DOI: 10.1080/10400435.2014.963258

Providing Assistive Technology Applications as a Service Through Cloud Computing

DAVIDE MULFARI*, ANTONIO CELESTI, MASSIMO VILLARI, and ANTONIO PULIAFITO

DICIEAMA, University of Messina, Messina, Italy

Users with disabilities interact with Personal Computers (PCs) using Assistive Technology (AT) software solutions. Such applications run on a PC that a person with a disability commonly uses. However the configuration of AT applications is not trivial at all, especially whenever the user needs to work on a PC that does not allow him/her to rely on his / her AT tools (e.g., at work, at university, in an Internet point). In this paper, we discuss how cloud computing provides a valid technological solution to enhance such a scenario.With the emergence of cloud computing, many applications are executed on top of virtual machines (VMs). Virtualization allows us to achieve a software implementation of a real computer able to execute a standard operating system and any kind of application. In this paper we propose to build personalized VMs running AT programs and settings. By using the remote desktop technology, our solution enables users to control their customized virtual desktop environment by means of an HTML5-based web interface running on any computer equipped with a browser, whenever they are.

Keywords: assistive technology, cloud, virtual machines, remote desktop

Introduction

Cloud computing is a model for enabling on demand network access to a shared pool of configurable computing resources that are rapidly provisioned and released with minimal management effort or service providers interaction (Mell & Grance, 2011). Among these resources, we focus on Virtual Machines (VMs) that can be considered as software implementations of a real computer able to execute an operating system and programs just like a physical machine does. By considering this scenario, if the VM runs a desktop environment, a user can interact with the available programs by using remote desktop client solutions.

In this article, we evaluate how such kind of VMs can support the usage of Assistive Technology (AT) applications intended for a traditional desktop environment. As an example, some users with low vision use screen readers as their primary means of accessing a computing environment; others prefer to execute screen magnification software solutions in order to better navigate a visual interface. People with dexterity impairments who cannot use a standard mouse or keyboard rely on several accessibility preferences available on widely used operating systems in order to adjust the way the computer responds to the mouse or keyboard input. In most cases, these AT tools are tailored applications installed on Personal Computers (PCs) a user with a disability commonly uses.

By using remote VMs, we propose to virtualize the overall desktop environment used by a person with a disability on his/her own computer, including the operating system (e.g., Microsoft Windows) and customized AT programs and settings. Additionally, our solution allows users to interact with tailored AT software programs by using an HTML5 web interface accessed from any networked physical PC equipped with a standard HTML5 web browser without having to set up any additional piece of software. The advantages of this approach are multiple (Mulfari, Celesti, Puliafito, & Villari, 2013). End users can successfully manage public computers (e.g., at an Internet point) that enable them to only work with a standard web browser: By using an Internet connection, disabled people can access their own personal virtual desktops from anywhere and whenever they want, and interact with their customized AT tools. In addition, AT experts can suitably customize remote VMs in order to support people with disabilities who live away from them, as in underdeveloped countries. A similar scenario may be related to the educational field in order to help disabled students attending a college. In the latter case, the student is able to access the VM from any shared networked computer (such as in a laboratory or in a library) without having to bring his/her own laptop to work. In addition, remote AT experts are able to meet the demand of persons with disabilities by adjusting desktop applications and AT software running on VMs. In such cases, users do not need an up-to-date hardware to interact with a working computing environment, since they can suitably work on a thin-client computer with just a web browser.

In order to evaluate the use of a traditional desktop environment with AT applications on remote virtual desktops, we discuss an implementation of a Cloud computing system based on the integration of Guacamole (i.e., an HTML5 remote desktop gateway), Virtual Network Computing (VNC) (i.e.,

*Address correspondence to: Davide Mulfari, DICIEAMA, Messina, Italy. Email: dmulfari@unime.it

a technology to control the remote desktop environments), Oracle VM VirtualBox (i.e., a software for controlling virtual environments; also called hypervisor), and CLEVER (i.e., a piece of middleware to build Infrastructure as a Service [IaaS] Cloud managing VMs; Celesti, Fazio, Villari, & Puliafito, 2013; Celesti, Tusa, Villari, & Puliafito, 2012).

The rest of this article is organized as follows. Related works are discussed in the next section. Then we discuss our software system; after that, we discuss how to configure our system in order to support the use of several AT tools running in VMs. In the subsequent section, we analyze the performance of our prototype, using an objective methodology. Lastly, conclusions of the article are drawn.

Related Works

In our opinion, AT in cloud computing is at the early stage. Several works exist in literature, but they do not deal with VMs, since their main objective is to make possible to surf the Internet for computer users with disabilities. Augmented browsing systems are intended to improve accessibility on any website (Mangiatordi & Sareen, 2011; Mirri, Salomoni, & Prandi, 2011) and they work on additional plugins in order to make on-the-fly changes to web page content after or before the same page is loaded in the browser window. Firefixia (de Santana, de Oliveira, Almeida, & Ito, 2013) is a web browser customization toolbar designed to support people with dyslexia by adapting the presentation of web content according to their preferences.

WebAnywhere is a remote web-based screen reader application allowing people who are blind to access the web from any networked computer with a modern web browser including self-voicing mechanisms (Bigham, Prince, & Ladner, 2008).

These AT web-based tools are mainly intended to allow the end user to access the Internet. Although functionality remains limited compared to equivalent desktop applications (which the end user can run on a remote VM), the major benefit of this approach is that it increases the accessibility of web sites when viewed on public machines in which users do not have permission to install custom software (Borodin, Bigham, Dausch, & Ramakrishnan, 2010).

Cloud4all is an international project funded by the 7th Framework Programme of the European Union that will advance the concept of the Global Public Inclusive Infrastructure (GPII) (Vanderheiden & Treviranus, 2011). Such an initiative is a synthesis of ideas and efforts of many people coming from several countries. It aims to provide a novel paradigm in accessibility (Vanderheiden, Treviranus, & Chourasia, 2013) by augmenting adaptation of individual products and services with automatic personalization of any mainstream product or service.

By comparison with the aforementioned solutions, in this work, we do not present a novel AT software solution for interacting with a computer, but we propose a new way to access existing AT tools by using cloud computing. To this end, we propose an open source software architecture able to achieve such a goal.

Software Architecture Description

Figure 1 shows the block diagram of our software architecture, highlighting the main components and their interaction.

End users work on a real computer equipped with any HTML5 web browser: By loading a specific web page, users are able to execute an HTML5 web application acting as a remote desktop client. This software is deployed in a web server, and it interacts with a remote desktop proxy (gateway) server that acts as an interface between the remote desktop server process running on a VM and the aforementioned HTML5 remote desktop client. Virtual environments run a common operating system with standard desktop applications, including AT programs. For example, a user who is blind can get his/her favorite screen reader on the personal VM in order to access it whenever he/she needs by means of any networked PC supporting a modern HTML5 web browser.

As depicted in Figure 1, the proposed software architecture can be split into two main subsystems: the Remote Desktop system and the IaaS Cloud system. In the following, we provide further details about such subsystems.

The IaaS Cloud System

In order to build the proposed infrastructure for using VMs, we need a software to arrange an IaaS distributed system.

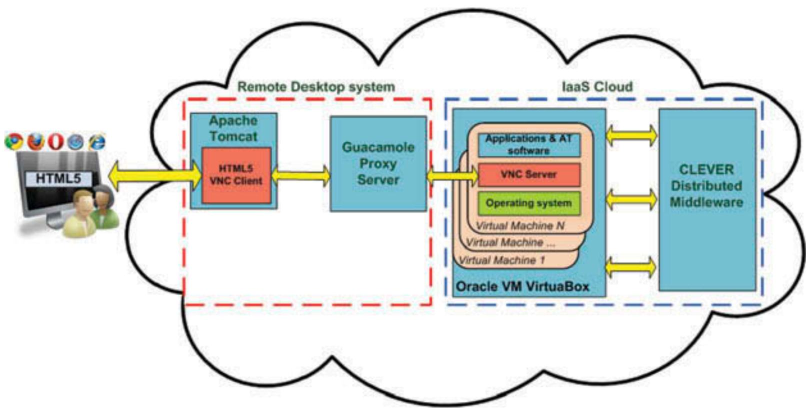


Fig. 1. Software architecture: The Remote Desktop and the IaaS Cloud systems.

Such a kind of software is referred to as Virtual Infrastructure Manager (VIM), and it has to be fault-tolerant, scalable, performing, secure, and federation-enabled. In this application scenario, we adopt CLEVER (Celesti, Tusa, et al., 2012), a modular, extensible, and interactive middleware fitting the aforementioned requirements. CLEVER allows us to control a Linux-based cluster environment in which several VMs can be dynamically managed. it basically performs the following operations:

- provide functions to start, shut-down, destroy, and migrate (move) VMs;
- monitor the VMs behavior and their performance in terms of central processing unit (CPU), memory, and storage;
- provide fault-tolerance features against computer hardware crashes and software failures;
- manage VM disk images;
- provide tools to create a new VM from an existing template including a given operating system (OS) and a set of desktop programs.

The CLEVER middleware is based on a distributed software system composed of a cluster of several nodes with a host level management module (Host Manager). A single node may also include a cluster level management module (Cluster Manager). All the entities interact exchanging information using XMPP (Extensible Messaging and Presence Protocol), that is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language). The set of data needed to enable the middleware functions is stored within a database deployed in a distributed fashion (Celesti, Tusa, et al., 2012).

CLEVER also supports various types of virtualization software packages (hypervisors) aimed at managing multiple VMs running on the host managers physical machines. The application discussed in the present article uses Oracle VM VirtualBox as hypervisor.

The Remote Desktop System

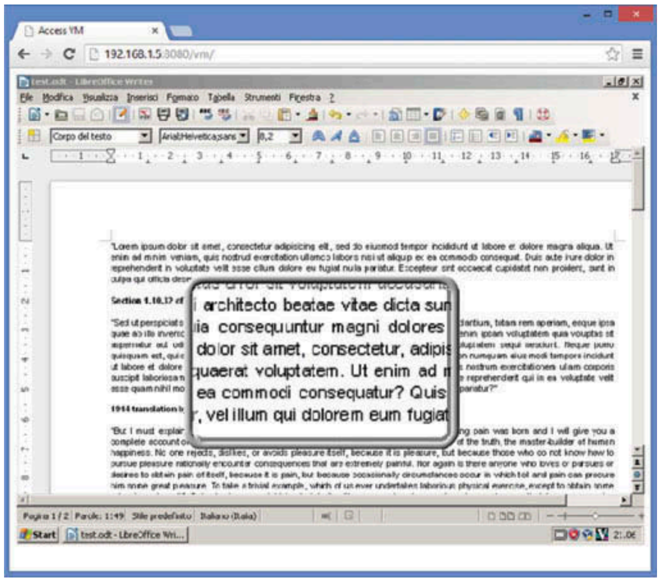
The system is responsible for accessing the desktop environment running on a VM through an HTML5 web interface. This approach requires no additional clients available on the user's physical computer. Thus, all the available software programs on a VM can be considered as a service over the network, eliminating the need to install additional pieces of software (or plugins) on the physical PC running the web browser.

In order to achieve such a goal, we used the VNC technology. It consists of an open source graphical desktop sharing system that uses the Remote Frame Buffer (RFB) protocol to remotely control a networked computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network. A standard VNC system is composed of three main elements: The VNC server is the software on the machine that shares its desktop environment, and it passively allows the client to take control of its software resources; the VNC client (or viewer) is the application that watches, controls, and interacts with the server. Finally, the VNC protocol (RFB) is very simple, based on basic graphic primitives from server to client and event messages from client to server.

To deploy such a kind of VNC system on the Cloud-based scenario previously described, we installed a VNC server on each virtual machine. Then by using an existing remote desktop proxy server, we setup a powerful software interface to translate VNC protocol for our HTML5 VNC web application. The latter software is deployed on a web server, and it works on an HTML5 Canvas tag to display the remote screen content within a custom web page; it uses specific JavaScript functions for handling mouse and keyboard events. An example is shown in Figure 2.

The main component of the Remote Desktop system is the *guacd* process, and it acts as Proxy Server. It consists of a Java piece of software that dynamically loads support for remote desktop protocols (RDP; called "client plugins") and connects them to remote desktops based on instructions received from the web application. The Guacamole proxy runs in the background, and it listens for Transmission Control Protocol (TCP) connections from the web application. guacd also does not understand any RDP, but rather implements just enough of the Guacamole protocol to determine which protocol support needs to be loaded and what arguments must be passed to it. Once a client plugin is loaded, it runs independently of guacd and has full control of the communication between itself and the web application until the client plugin terminates. For these reasons, the HTML5 VNC client software has been deployed on a servlet container, such as Apache Tomcat.

While such web-based VNC implementations allow end user to view the virtual desktop environment within a web page and to interact with it by means of mouse and keyboard, the major drawback of the solution consists in the lack of the remote audio redirection feature, so you cannot hear audio signals produced by the desktop applications running on the web-accessed virtual machine. In order to address this issue, we propose a custom technical solution: According to the VirtualBox setup, each VM supports a virtual sound card, and the audio output is managed


Access VM
192.168.1.5:8080/vm/
test.odt - LibreOffice Writer
File Modifica Visualizza Inserisci Formato Tabella Strumenti Finestra ?
Corpo del testo
Arial;Helvetica;sans
8,2
Pagina 1 / 2 Parole: 1:49 Stile predefinito Italiano (Italia)
Start
test.odt - LibreOffice Wri...

i architecto beatae vitae dicta sun
ia consequuntur magni dolores
dolor sit amet, consectetur, adipis
quaerat voluptatem. Ut enim ad n
ea commodi consequatur? Quis
r, vel illum qui dolorem eum fugiat

Fig. 2. A screen magnifier running on a VM accessed via a web browser.

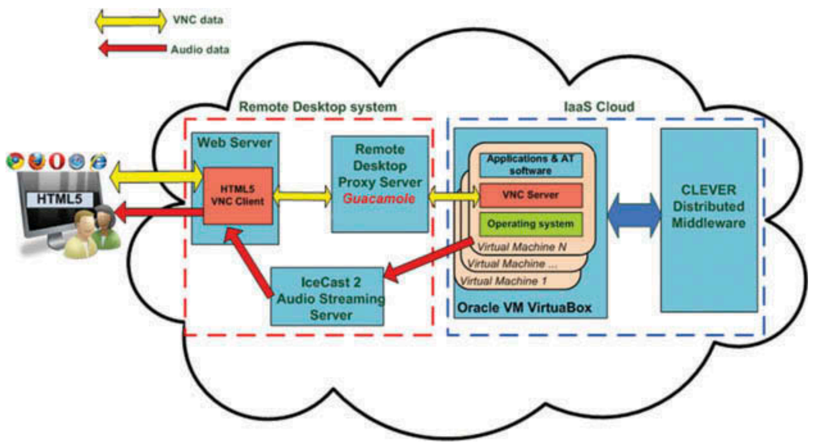


Fig. 3. Block diagram of the proposed implementation including audio redirection process.

by the PulseAudio subsystem running on the Linux physical server running the hypervisor, such as an Host Manager. The basic idea is to compress the audio signal in order to send it to the IceCast2 audio streaming server. Such operation requires GStreamer pipelines and Lame MP3 encoding tools. Finally, the same streaming server handles the audio outputs and transmits them over the network. Therefore the whole VNC-based system can be depicted as shown below (Figure 3).

Such a software infrastructure works theoretically, but, depending on the circumstances, a text-to-speech application (e.g., a screen reader) running on a virtual machine could experience unacceptable delays. As shown in the next section, this user experience, in fact, may result in a high lag time from when the person performs an input action (i.e., he/she presses a key) until he/she hears the voice output.

Additional Support for AT Preferences

In order to enhance the accessibility of desktop applications running on web-accessed VMs, the HTML5-based VNC client includes specific features that meet the demands of people with dexterity impairments who do not use a standard mouse or keyboard to access a computer. Typically these users rely on several keyboard assistive preferences in order to emulate mouse actions and to manage keyboard input so that key combinations are easier to press, typing is easier, and accidental keystrokes are ignored.

To implement such functionalities on the remote desktop web client, some JavaScript functions have been used in order to detect any low-level keyboard event occurring on the HTML5 Canvas tag and to interact with the remote desktop proxy server properly. For this purpose, these pieces of software act as a filter for mouse and keyboard events: It converts the keystrokes into the related input actions for the controlled virtual machine.

By using Guacamole as gateway solution, in the following code (Listing 1), we show how to interface a numeric keypad in order to move the mouse pointer on the virtual desktop environment.

To emulate mouse movements and click actions by means of the numeric keypad, we have worked on the Guacamole Application Programming Interface (APIs). Keyboard events are abstracted in Guacamole. The Keyboard object has only keyup and keydown events; thus we used the latter event to detect if the user presses a key of the keypad, as shown in the source code. Each key has a specific keycode, and we can detect it by using the if statement (lines 7–11). We can handle two kinds of events: If the user presses a certain key (lines 18–22), we perform a click action, else we work on the mouse object to move the pointer on the remote web-accessed VM. Such a mechanism allows us to avoid any keyboard configuration on the physical machine used by the person with disability, Furthermore, the discussed implementation allows us to replace the numeric keypad by any human interface device (HID) acting as a standard keyboard or mouse.

Performance Evaluation

In this section, we discuss a practical approach to investigate the usability of desktop applications running on a remote virtual

```
 1 var keyboard = new Guacamole.Keyboard(document);
 2 var statomouse = new Guacamole.Mouse.State
 3                      (0,0,false,false,false,false,false);
 4 var px = 0;
 5 var py = 0;
 6 keyboard.onkeydown = function (keysym) {
 7 if (keysym>65360){
 8     if (keysym==65362) py = py - 2;
 9     else if (keysym==65364) py = py + 2;
10     else if (keysym==65363) px = px + 2;
11     else if (keysym==65361) px = px - 2;
12     if (px < 0) px = 0;
13     else if (px > 795) px = 795;
14     if (py < 0) py = 0;
15     else if (py > 595) py = 595;
16     statomouse.y = py;
17     statomouse.x = px;
18     if (keysym==65367) statomouse.left = true;
19     guac.sendMouseState(statomouse);
20     if (keysym==65367){
21         statomouse.left = false;
22     }
23     guac.sendMouseState(statomouse);
24 }
25 else
26     guac.sendKeyEvent(1, keysym);
27 };
```

Listing 1. JavaScript function to simulate mouse events by using the numeric keypad.

```
from pymouse import PyMouse; import time

sleepsec=10

cnt=51

m=PyMouse()

x, y = m.position()

for n in range(1,cnt):

    m.click(x,y,1)

    time.sleep(sleepsec)
```

Listing 2. Python program to simulate click events.

machine accessed by the end user by means of the HTML5-based VNC client web application. We aim to evaluate how the usage of the web client makes possible an effective interaction with two types of programs on virtual desktop environments: applications working with an audio streaming output and pieces of software which use the computer's screen as output channel. The proposed analysis is mainly aimed at calculating the overall lag time which the user experiences by using the HTML5 remote desktop client. Our evaluation does not relate to a subjective real-world user-experience concerning the usage of AT computer applications (like screen readers); we aim to estimate an objective parameter able to characterize a potential user-experience when working with programs executing on a web-accessed VM.

For this reason, we decided to simulate an end user working on a real computer (called client node PC) that runs Ubuntu Desktop 12.04 as operating system, with Google Chrome as web browser for accessing a given virtual machine. In this scenario, we consider a single mouse click on the HTML5 Canvas tag displaying the virtual machine's screen content as an input event, whereas an audio playback and a given screen update are recognized as output response provided by the software infrastructure we intend to evaluate. For convenience, the client node PC executes the following Python script to perform a given number of single mouse click events at given time intervals:

Additionally, the client node PC runs Wireshark as the network protocol analyzer in order to monitor the traffic produced by the web application running in the web browser. By looking at the high level software architecture depicted in Figure 1, we are able to monitor all the network packets between the HTML5 remote desktop client and the Guacamole remote desktop proxy server.

Testbed Design

To perform the planned experiments, the proposed software infrastructure was arranged in a distributed fashion. Software modules constituting the Remote Desktop system (Apache Web Server, Tomcat 6 extension, Guacamole remote desktop proxy/gateway) were deployed on an IBM LS20 server blade with Ubuntu Server 12.04 64-bit as operating system. The overall Iaas Cloud system, including the CLEVER middleware, was deployed on a different IBM LS20 server blade running Ubuntu Server 12.04 64-bit operating system and Oracle VM VirtualBox 4 as hypervisor. Such two physical server machines are connected to each other by means of a Gigabit Ethernet connection. Also, we created a single virtual machine instance supporting Microsoft Windows 7 Ultimate and configured the environment to execute UltraVNC as the VNC server.

All the experiments were performed by considering two network scenarios; in the first scenario, the client node computer accesses the given virtual machine through a Gigabit Ethernet LAN (Local Area Network) connection, and the second network scenario provides a 7 Mbps Digital Subscriber Line (DSL) connection to access the aforementioned virtual environment. Experiments conducted in an Intranet scenario were useful for evaluating the usability of the proposed cloud architecture, for example, in the private cloud of an university, government, society, and so on. In such environments, we assume that the network latency is negligible. This assumption is reasonable if we assume that VMs can be transferred inside the private Cloud through VM migration techniques (Celesti, Fazio, Villari, & Puliafito, 2012; Celesti, Tusa, Villari, & Puliafito, 2010). Instead, experiments conducted in the Internet scenario have been useful for understanding the behavior of the prototype in a public Cloud.

Working with VNC Video Frame

In order to numerically estimate the delay time in displaying (on the Canvas tag) an entire video frame, the virtual machine runs a custom Java desktop application. It works on an empty 800 × 600 GUI, and it listens for single mouse click actions. When such an event occurs, the software changes the background color of the form element; then, a separate thread begins its execution—it sleeps for two seconds and modifies the color of the graphical user interface (GUI). Then the aforementioned thread sleeps for two seconds again, and then it updates the background color and stops its execution.

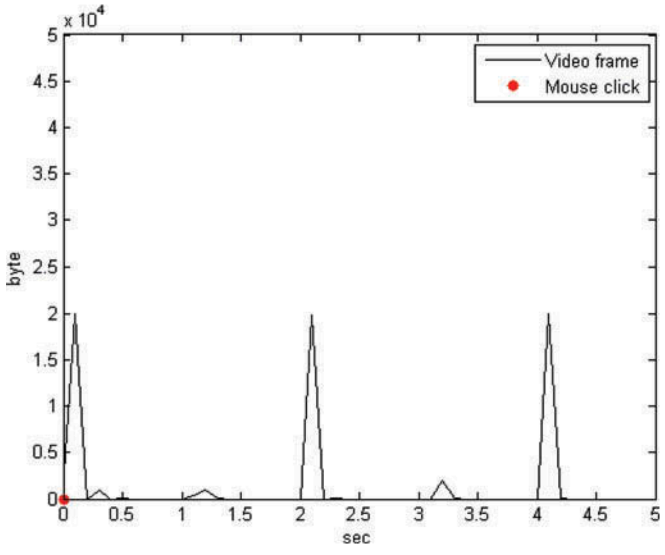


Fig. 4. Throughput related to video refresh and mouse click event.

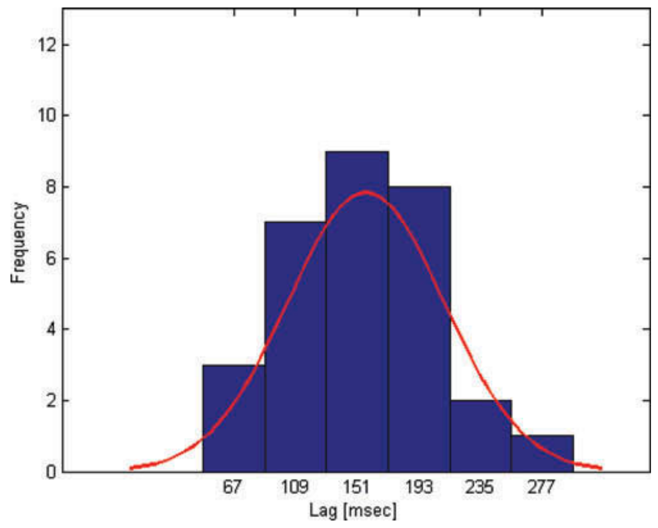


Fig. 5. Frequency histogram with the normal distribution concerning the lag due to a screen update over a LAN connection.

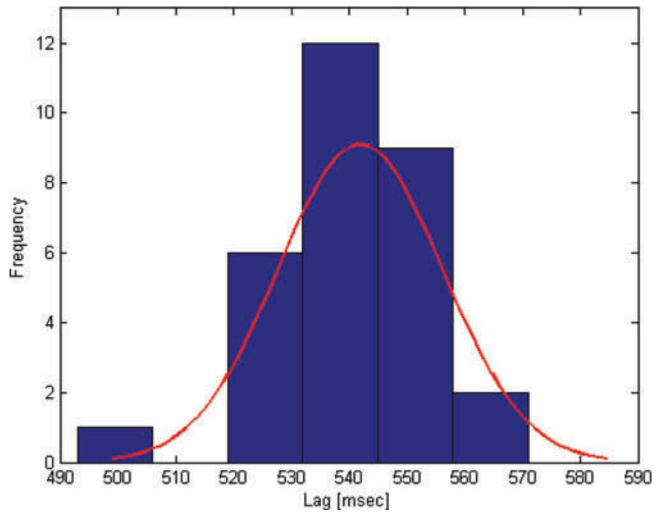


Fig. 6. Frequency histogram with the normal distribution concerning the lag due to a screen update over a DSL connection on a weekend.

By using Wireshark tools, we analyzed the throughput graph concerning the experience discussed above: We observed that the update of the background color produces notable spikes on the aforementioned graph, as shown in Figure 5. It refers to the throughput related to the HTML5 web application while the custom Java program is running on the virtual machine. For convenience, as can be seen on the graph, a single mouse click event was performed at $t = 0$ sec. Therefore, the Java application starts its execution, as mentioned above. In this scenario, we are also able to get the timestamp value related to the color update. So, we can calculate the delay time discussed in this Section.

The discussed experience was repeated several times, and the results are depicted in the following distribution graph (Figure 5). It shows both the histogram and the idealized shape of the normal distribution with the mean and the standard deviation evaluated at the given data set.

The bars in the graph indicate how many times each value occurred in the data set, whereas the observed values concerning the lag due to a video refresh are placed in the x-axis. Here, the tallest bar highlights the value that occurs more often and the shape of the distribution gives us information about the distribution of the data around the aforementioned value. We can also appreciate a 157 ms mean delay time using the LAN connection mentioned above. In relation to AT software tools, this feature may be useful to investigate the usage of several kinds of screen magnification software, which respond to mouse input actions by enlarging some parts of the computer's screen. By considering our LAN network scenario, a screen magnification tool seems to be usable.

We repeated the aforementioned experiments by working with our 7 Mbps DSL connection. In the latter case, we detected a higher lag time due to network traffic. Experiments were conducted on a weekend. The VM responded to the same single mouse input in a more variable delay time, as depicted in Figure 6.

This may cause troubles using a screen magnification application running on the web-accessed VMs.

The depicted poor performances may be related to several reasons, such as:

- inability to optimize the behavior of all the software components;
- the software elements cannot be suitable to pursue our goals;
- the VNC remote desktop technology does not allow to gain a better user experience.

Working With Audio Stream

In order to evaluate the lag time regarding a remote audio playback, the virtual desktop plays a 2 s WAV file periodically. For convenience, the VM runs the VLC Media Player as background process, and it responds to a single mouse click by playing the audio file. By using the audio redirection infrastructure, the live streaming was played on the client node's speakers via the proposed web application. Also, this physical host runs the mouse daemon process in order to perform single click events periodically. Therefore, by using Wireshark on the client node, we are able to analyze the network traffic between the machine and the IceCast server, while the HTML5 application works on the aforementioned live streaming. As described earlier, we evaluated the throughput graph related to this experience, as shown in Figure 7.

In this case, the audio pipeline produces a background noise with notable spikes when the web application loads the WAV file. So, by analyzing the timestamp values of such network packets, we can numerically evaluate the lag time in playing an audio streaming. As depicted in Figure 8, we represented the data set using the histogram and the normal distribution with the mean and the standard deviation evaluated at the aforementioned data set.

In this case, on the x-axis, one can view the values concerning the lag due to a remote audio, and we appreciate a 1384 ms. mean delay time using the LAN connection.

Therefore, an interactive audio application (such as a screen reader) is not usable by the end user. Such results can be confirmed by considering the DSL network scenario and the network

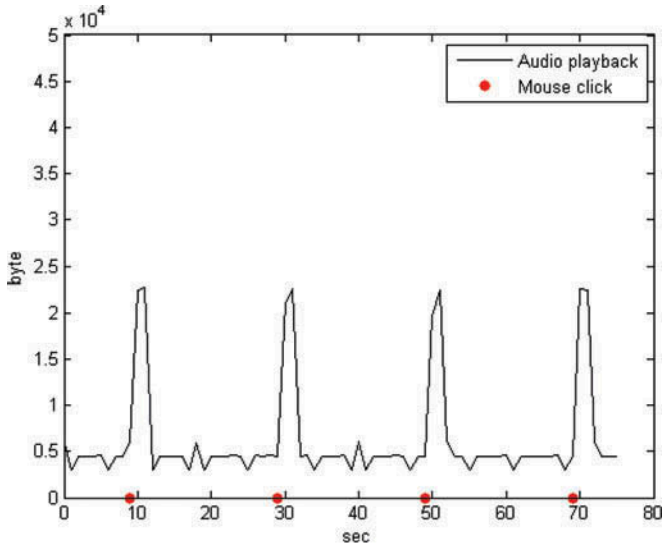


Fig. 7. Throughput related to remote audio playback with mouse click event.

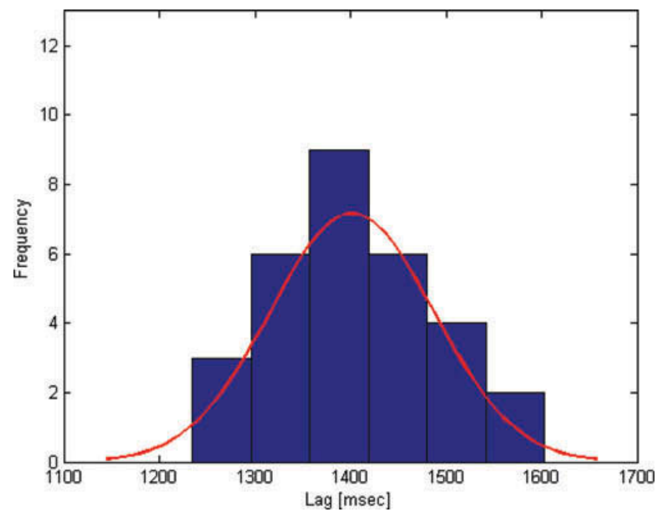


Fig. 8. Frequency histogram with the normal distribution concerning the lag due to an audio playback over a LAN connection.

traffic on a weekend. Here, we assume a variable network latency, so the overall lag time is higher compared to the previous case.

There is still a long way to go toward the achievement of an assistive usable solution based on the software architecture discussed in this article. Future research directions will aim to improve performances, and they will deal with the following topics:

- optimize the current software architecture;
- review all the software components;
- evaluate if a different technology, such as RDP can work better than VNC.

We believe that the proposed solution highlights a new approach to provide AT applications as a service through cloud computing. It has great potentialities and represents a valid solution to simplify the access to IT services to disabled people. The

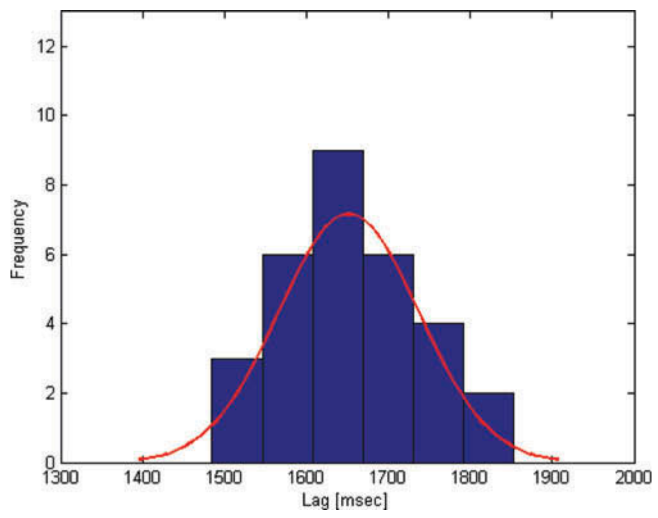


Fig. 9. Frequency histogram with the normal distribution concerning the lag due to an audio playback over a DSL connection on a weekend.

approach is very promising, and further investigations are needed to tune and optimize the overall performance.

Conclusions

The present article does not deal with a novel AT software solution for interacting with computers. We propose a solution intended for people with disabilities who are used to execute tailored AT tool to manage their own PCs.

In such a scenario, we suggest to run these desktop applications on VMs and access them via a web interface. By using any networked computer with a standard web-browser, users with disabilities can manage anywhere their usual desktop environment with accessibility tools, without having to install new software on the physical PC. To provide such services, an open-source software architecture was designed. The working prototype exploits the VNC technology to remotely control VMs, whereas a web-based remote desktop solution enables users to interact with their applications running on VMs. This solution was adjusted to support keyboard accessibility tools and remote audio redirection, but the effective use of the whole architecture resulted in poor performance. To conclude, we can say that this article addresses a new line of research with very challenging opportunities, where lots of technical work still need to be done.

In future works, we plan to test the RDP to access VMs by means of a web interface. Such a technology comes with the remote audio redirection, and it should improve network bandwidth usage compared to a VNC connection. In order to manage RDP connections on remote VMs, we plan to use the VirtualBox Remote Display Protocol (VRDP), allowing us to use any standard RDP client to control the remote virtual desktop environments.

Nowadays, cloud computing technology may really revolutionize the delivery of AT services. For this purpose, we work for the design of a Community for Cloud AT Software as a service (SaaS). According to the NIST Definition of Cloud computing

(Mell & Grance, 2011), a Community Cloud is a cloud service model that provides a solution to a limited number of individuals or organizations that have shared concerns. Especially, we plan to implement the proposed software architecture in the cloud platform at the University of Messina. Specifically, we intend to equip VMs with tools and services for students with disabilities. In this way, each student will be able to access his/her personal VM by using any computer in the campus. Moreover, AT experts will be able to support many students by adjusting several remote virtual desktop environments.

Funding

The research leading to the results presented in this article has received funding from the Project "Design and Implementation of a Community Cloud Platform aimed at SaaS services for on-demand Assistive Technology".

References

Bigham, J. P., Prince, C. M., & Ladner, R. E. (2008). Webanywhere: Enabling a screen reading interface for the web on any computer. In *Proceedings of the 17th International Conference on World Wide Web. ACM*

Borodin, Y., Bigham, J. P., Dausch, G., & Ramakrishnan, I. V. (2010). More than meets the eye: A survey of screen-reader browsing strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (w4a)* (pp. 13: 1–13:10). New York, NY: ACM. doi:10.1145/1805986.1806005

Celesti, A., Fazio, M., Villari, M., & Puliafito, A. (2012). Virtual machine provisioning through satellite communications in federated cloud environments. *Future Generation Computer Systems*, *28*(1), 85–93. doi:10.1016/j.future.2011.05.021

Celesti, A., Fazio, M., Villari, M., & Puliafito, A. (2013). Se clever: A secure message oriented middleware for cloud federation. In *IEEE Symposium on Computers and Communications (ISCC)*. IEEE Computer Society.

Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2010). Improving virtual machine migration in federated cloud environments. In *Second International Conference on Evolving Internet (Internet)* (pp. 61–67). doi:10.1109/INTERNET.2010.20

Celesti, A., Tusa, F., Villari, M., & Puliafito, A. (2012). Integration of clever clouds with third party software systems through a rest web service interface. In *Ieee Symposium on Computers and Communications (iscc)* (pp. 827–832). IEEE Computer Society.

De Santana, V. F., de Oliveira, R., Almeida, L. D. A., & Ito, M. (2013). Firefixia: An accessibility web browser customization toolbar for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility* (pp. 16: 1–16:4). New York, NY: ACM. doi:10.1145/2461121.2461137

Mangiatordi, A., & Sareen, H. S. (2011). Farfalla project: Browser-based accessibility solutions. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility* (pp. 21: 1–21:2). New York, NY: ACM. doi:10.1145/1969289.1969317

Mell, P. M., & Grance, T. (2011). *Sp 800-145. The NIST definition of cloud computing* (Technical Report). Gaithersburg, MD.

Mirri, S., Salomoni, P., & Prandi, C. (2011). Augment browsing and standard profiling for enhancing web accessibility. *In Proceedings of the International Cross-Disciplinary Conference on Web Accessibility* (pp. 5: 1–5:10). New York, NY: ACM. doi:10.1145/1969289.1969297

Mulfari, D., Celesti, A., Puliafito, A., & Villari, M. (2013). How cloud computing can support on-demand assistive services. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility* (pp. 27: 1–27:4). New York, NY: ACM. doi:10.1145/2461121.2461140

Mulfari, D., Celesti, A., Villari, M., & Puliafito, A. (2013, Dec). Using virtualization and Guacamole/VNC to provide adaptive user interfaces to disabled people in cloud computing. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)* (pp. 72–79). doi:10.1109/UIC-ATC.2013.42

Vanderheiden, G., & Treviranus, J. (2011). Creating a global public inclusive infrastructure. In C. Stephanidis (Ed.), *Universal Access in Human-Computer Interaction. Design for All and eInclusion* (Vol. 6765, pp. 517–526). Berlin, Germany: Springer Berlin Heidelberg. doi:10.1007/978-3-642-21672-557

Vanderheiden, G. C., Treviranus, J., & Chourasia, A. (2013). The global public inclusive infrastructure (GPII). In *Proceedings of the 15th International ACM Sigaccess Conference on Computers and Accessibility* (pp. 70: 1–70:3). New York, NY: ACM. doi:10.1145/2513383.2513395