A Platform Agnostic Remote Desktop System for Screen Reading

Syed Masum Billah Vikas Ashok Donald E. Porter IV Ramakrishnan
Stony Brook University
{sbillah, vganjiguntea, porter, ram}@cs.stonybrook.edu

ABSTRACT

Remote desktop technology, the enabler of access to applications hosted on remote hosts, relies primarily on scraping the pixels on the remote screen and redrawing them as a simple bitmap on the client's local screen. Such a technology will simply not work with screen readers since the latter are innately tied to reading text. Since screen readers are locked-in to a specific OS platform, extant solutions that enable remote access with screen readers such as NVDARemote and JAWS Tandem require homogeneity of OS platforms at both the client and remote sites. This demo will present Sinter, a system that eliminates this requirement. With Sinter, a blind Mac user, for example, can now access a remote Windows application with VoiceOver, a scenario heretofore not possible.

1. INTRODUCTION

Remote desktop technology is primarily used to access applications that are hosted on remote machines, such as a physician accessing a medical records application running on an office server from her home. The graphical display of the remote system is virtualized and shipped across the network to the client. Virtualization emulates the graphics card frame buffer in which the pixel values are scraped from the screen of the remote system, and redrawn as a simple bitmap on the client's local screen. This approach yields seamless access to remote applications, as if they were running on the local desktop. Remote desktop access is indispensable to users engaged in activities such as telecommuting, distance learning, remote troubleshooting and maintenance.

Unfortunately, remote desktop technology in its current incarnation, namely via emulation of the graphics frame buffer, simply will not work for blind users. These users rely on screen readers to interact with digital content. Screen readers require semantic information, such as text and hierarchical relationships to narrate screen contents—all of this information is lost by the time the screen is rendered as a bitmap.

ASSETS '16 October 23-26, 2016, Reno, NV, USA
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4124-0/16/10.
DOI: http://dx.doi.org/10.1145/2982142.2982151

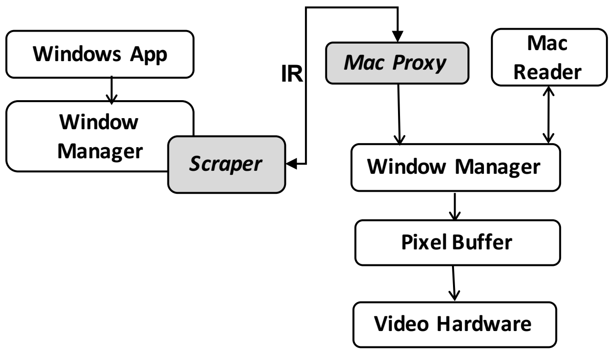


Figure 1: A schematic of remote access in Sinter.

One obvious solution is to run a screen reader on the remote host and relay the synthesized audio to the client. But network delays, especially over WANs, can introduce unacceptable latencies to relaying audio, a fact validated by our experiments [1]. A second approach, exemplified by commercial systems such as NVDARemote [3] and JAWS-Tandem [2], is to intercept text from the remote screen reader just before audio synthesis, relay this text to the local client and synthesize audio locally. Getting this to work requires identical screen readers, one running remotely and the other locally. The serious problem with such a solution is that screen readers are locked into a single operating system platform; the differences in the underlying accessibility APIs (e.g., Microsoft's MSAA and UI Automation, Apple's Accessibility, and GNOME's ATK & AT-SPI) has been a strong barrier to portability. Consequently, NVDARemote-like solutions require both the remote and local platforms to be similar. Platform-specific remote access solutions are clearly inadequate in the modern computing era where computer users increasingly use applications designed for different OSes, spanning desktops, laptops, tablets, mobile devices, cloud and other platforms. For instance, a Mac user may access a cloud-based Windows remote desktop to run an application required for her job. Similarly, a Windows desktop user may use VMware workstation to develop and test a Linux server application. Such scenarios are the norm these days and, thus, calls for platform-independent remote access solutions for screen reader users. Our Sinter system [1] provides such a solution.

2. *SINTER* SYSTEM OVERVIEW

Sinter is predicated on the observation that applications on every OS consist of similar User Interface (UI) widgets such as buttons, drop-down menus, text fields, etc. The key idea then is to create a UI model of the application's GUI from the existing, platform-specific accessibility APIs, analogous to an HTML document object model (DOM) tree, convert the model into a generic intermediate representation (IR), and finally render the IR encoding of the application GUI on a different platform, using native UI widgets. A screen reader on the client system can then read the native rendering of the IR. Fig. 1 is a high level schematic of this idea. The figure corresponds to a scenario where a user wants to run VoiceOver, the Mac screen reader, to read a Windows application on a remote system. In Sinter, a scraper on Windows mines the Windows-specific UI model of the application, and converts it into Sinter's IR which is shipped to a client system–Mac in this example, where a proxy converts the IR back to a native representation of these elements that VoiceOver understands. The Sinter proxy relays keystrokes and other user inputs back to the scraper, and the scraper relays incremental changes in the UI back to the proxy. In Sinter, the IR is a generic XML which we found to be sufficiently expressive for encoding a majority of standard UI element types on Windows and Mac. Having an IR opens up powerful opportunities for personalizing or making application specific modifications of the UI itself. For example, we made a mega-ribbon for Word in Fig. 2, which avoids cumbersome ribbon navigation for frequently-used buttons.

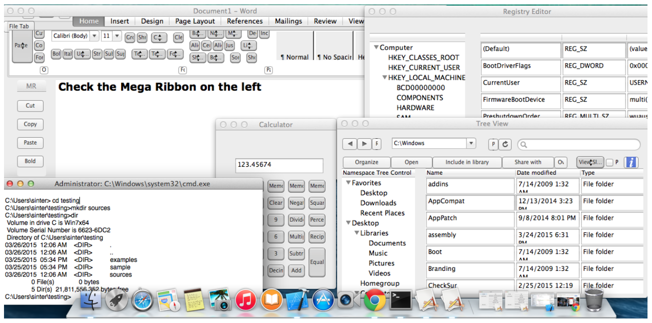
3. CURRENT STATUS

The Sinter idea and the engineering that went into building the system was presented in the 2016 European Systems Conference [1]. Since then, we have completed the implementation of a fully functional Sinter system. Furthermore, following IRB approval we also conducted a preliminary user study with 21 blind subjects at Lighthouse Guild. Currently, Sinter enables to screen-read the remote Windows application locally on a Mac with VoiceOver, and conversely, remote Mac application locally on Windows with JAWS and NVDA. Windows applications that can be accessed remotely from Mac clients include Microsoft Word, Windows Calculator, Windows Explorer, the Windows registry editor, and the DOS command line (command.exe) while remote Mac applications accessed from Windows clients include Apple Mail, HandBrake (a media ripping and transcoding utility), Messages, Calculator, and Contacts. Fig. 2 presents snapshots of Windows applications accessed remotely on an Mac client with Sinter.

A Sinter demo, which will be the first of its kind to the accessibility community, will exercise the functionalities of Sinter with these applications in place.

4. PRELIMINARY USER STUDY

21 blind subjects participated in a preliminary study primarily primed to get an initial assessment of Sinter, and gather additional requirements from users. Towards that they were given tasks to calculate with a remote calculator, and edit documents using Notepad running remotely. A System Usability Scale (SUS) questionnaire was administered at the end of the study relating to usability, ease of use, learning curve, etc. Sinter averaged a SUS score of around 78 which is considered good from a usability perspective. More importantly, a noteworthy highlight that emerged from the study was that users very much liked the fact that they can access remote applications using their preferred screen reader with customized settings. This eliminates the need to learn new screen readers or platforms, and thus allows them to stay in their own comfort zone. They also felt Sinter had the potential to be a game changer since it does not need a screen reader to be installed on a remote host which is often the situation they encounter in practice.



Check the Mega Ribbon on the left

123.45674

Administrator: C:\Windows\system32\cmd.exe

```
C:\Users\sinter> cd testing
C:\Users\sinter\testing>mkdir sources
C:\Users\sinter\testing>dir
 Volume in drive C is Win7x64
 Volume Serial Number is 6623-6DC2
 Directory of C:\Users\sinter\testing
03/26/2015  12:06 AM    <DIR>          .
03/26/2015  12:06 AM    <DIR>          ..
03/25/2015  05:34 PM    <DIR>          examples
03/25/2015  05:34 PM    <DIR>          sample
03/26/2015  12:06 AM    <DIR>          sources
               0 File(s)              0 bytes
               5 Dir(s)  21,811,556,352 bytes free
C:\Users\sinter\testing>
```

Computer
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
BCD00000000
COMPONENTS
HARDWARE

(Default)	REG_SZ	(value
BootDriverFlags	REG_DWORD	0x000
CurrentUser	REG_SZ	USERN
FirmwareBootDevice	REG_SZ	multi(
PreshutdownOrder	REG_MULTI_SZ	wuaus

Name	Date modified	Type
addins	7/14/2009 1:32 AM	File folder
AppCompat	12/13/2014 3:23 PM	File folder
AppPatch	9/8/2014 8:01 PM	File folder
assembly	3/24/2015 6:31 PM	File folder
Boot	7/14/2009 1:32 AM	File folder
Branding	7/14/2009 1:32 AM	File folder
CheckSur	2/25/2015 12:19	File folder

Figure 2: Clockwise from the top left: Microsoft Word, the Windows registry editor (regedit) Windows Explorer, Windows Calculator, and the Windows command line (cmd.exe). Word also displays the mega ribbon on the left hand side, automatically saving the most frequently used buttons.

5. DISCUSSION

Sinter requires only two simple conversions for each OS: one from the native accessibility API to the IR, and one from the IR back to the native accessibility API. Our experience with Sinter has been that, if one can reuse native libraries, this translation can be done in a few thousand lines of code in each direction. As a point of comparison, the NVDA Windows screen reader is over 50,000 lines of code. We found that IR transformation is a potent meta-programming tool for making accessibility enhancements that are desirable but missing in the original application. Their full potential remains to be explored. Finally, an extensive user study examining important aspects of Sinter such as impact of platform independence, and accessibility enhancing IR transformations is a task that we will undertake in the near future.

Acknowledgments: This research was supported in part by NSF grants IIS-1218570, IIS-1447549, CNS-1149229, CNS-1161541, CNS-1228839, CNS-1405641, CNS-1408695, CNS-1526707, and VMware.

6. REFERENCES

[1] S. M. Billah, D. E. Porter, and I. V. Ramakrishnan. Sinter: Low-bandwidth remote access for the visually-impaired. In *Proceedings of the Eleventh European Conference on Computer Systems*, pages 27:1–27:16, New York, NY, USA, 2016. ACM.

[2] Freedom Scientific. Jaws tandem quick start guide. http://www.freedomscientific.com/JAWSHq/JAWSTandemQuickStart. [Online; accessed 11-Jun-2016].

[3] NVDA. NVDA Remote brings free remote access to the blind. http://nvdaremote.com/.