Keeping Emulation Environments Portable

FP7-ICT-231954

How to

Run emulators remotely via the Emulation Framework

Date	23 February 2012
Document version	1.0
Author(s)	Maïté Braud (Tessella) Bram Lohman (Tessella) Jeffrey van der Hoeven (National Library of the Netherlands)
Organisation(s)	KEEP project Tessella National Library of the Netherlands

Executive Summary

This document summarises a short study performed by the KEEP project into remote emulation. In this context, remote emulation means: to run and control emulators over a network where the emulator runs on a server and the interaction with the emulator takes place on a client machine.

The Emulation Framework (EF) developed by the KEEP project does not support remote emulation out of the box. Although software and emulators can safely be stored remotely as a standalone web service by the EF, the actual execution of emulation processes takes place on the client machine. To take away the workload from the client computer, the EF can be altered to run as a remote application on the server using existing tools.

This study looked into several existing solutions to apply remote emulation. Based on a set of business and technical requirements (most importantly cost, performance, security and scalability) various technical possibilities have been researched:

- Virtual Network Protocol (VNC): RealVNC, TightVNC, UltraVNC, X11VNC, Guacamole;
- Remote Desktop Protocol (RDP): Remote Desktop Service, NX software, XRDP;
- GRATE: prototype for remote emulation developed by the PLANETS project;
- Combination of VNC server and Virtual Machine;
- Embedded VNC server in the emulator: VNC on application level;
- Citrix XenApp;
- Oracle Secure Global Desktop.

To demonstrate that remote emulation can work using the EF, a proof of concept has been carried out that integrated the EF with Tessella's Safety Deposit Box archiving solution. To handle the client/server interfacing for remote emulation, Guacamole was used displaying an emulator inside a web browser. This solution worked well but is unfortunately not (yet) viable in a production environment most importantly because Guacamole shares the server's entire desktop giving the end user complete access to the server.

Conclusion

Many open source and proprietary solutions are available for connecting to a computer desktop or application remotely via a network. However, there is no perfect solution as almost each solution in this report has a disadvantage regarding the before mentioned business and technical requirements (e.g. cost, performance or multi-platform support). Ultimately each organisation might have to choose its own specific solution and implement it the way it fits the organisation.

A table summarising the various solutions and their rating on costs, performance and security is given at the end of this document. This can be used to select the most appropriate solution for your organisation.

Table of Contents

1 Introduction

1.1 Purpose of this document

This document summarises a short study performed by the KEEP project into remote emulation. In this context, remote emulation means: to run and control emulators over a network where the emulator runs on a server and the interaction with the emulator takes place on a client machine.

The Emulation Framework (EF) is an open source software package developed by the KEEP project. It offers users a user-friendly way to access old digital computer environments and files. The EF configures and launches existing emulators automatically which takes away any difficulties for novice users to use complex emulation tools. The emulators and software packages used for recreating a particular computer environment are kept in a software archive and emulator archive. These archives can run on a remote server. However, the workload of running an emulation environment takes place on client side. This requires that the client machine is strong enough to run these kind of heavy computing processes. Figure 1 shows the default client/server setup of the EF (note that it is also possible to run all components of the EF at client side).

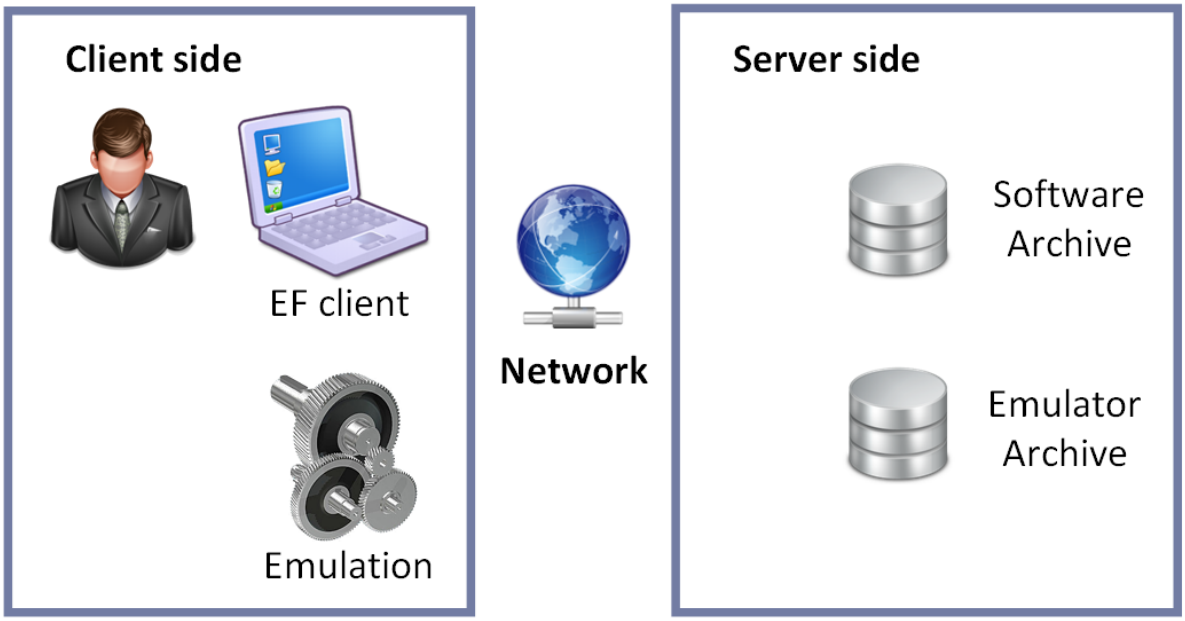

Figure 1: client/server setup of Emulation Framework

When the EF is used by an individual, this setup is probably not a problem. However, if the EF is applied in an environment where it serves as a public access tool for a larger audience all client machines are required to have the same powerful hardware specifications. This might be a showstopper in organisations where thin client provisioning is the policy or budgets do not allow heavy client machines. Therefore, it would be interesting to run the EF in a client/server setup in which the emulation process itself runs remotely and the interface is distributed to the client via a network. Figure 2 shows how this setup could look like.

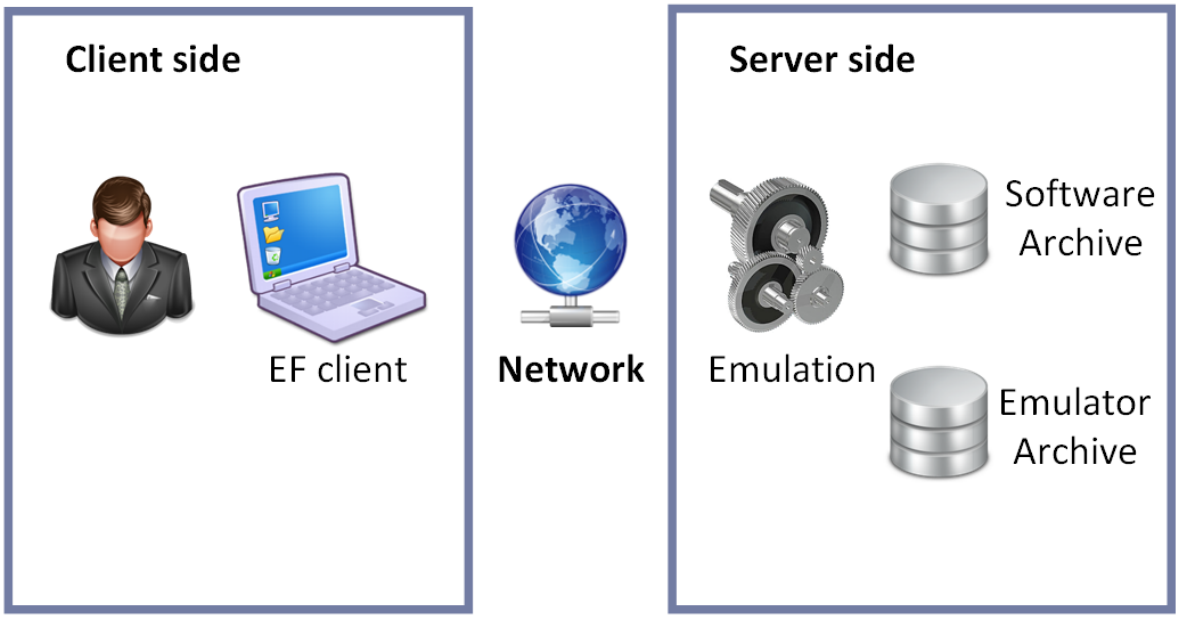

Figure 2: client/server setup of the Emulation Framework with remote emulation

The EF does not support remote emulation out of the box, but there are existing solutions available that can be applied to the EF which makes remote emulation possible. This study gives an overview of the most relevant solutions by hand. As the EF is open source, any of these solutions could be applied. As business needs and technical requirements are different for each organisation, this document will not recommend one specific solution but rather focuses on the possibilities currently available.

1.2 Scope

This report is not a technical note; it does not give you any indication of how to install/use a given technology.

It also does not provide specific cost information because this information is often not available or it is dependent on the customer (commercial or public organisation) and its specific configuration (how many licences are required, which server is installed, etc.). However, where possible a relative cost comparison is given.

1.3 About the Emulation Framework

The Emulation Framework (EF) allows you to render digital files and computer programmes in their native environment. This offers you the potential to view these files in their intended 'look and feel', independent from current state of the art computer systems. The spectrum of potential computer platforms and applications that can be supported is practically unlimited. The EF has build-in support for emulating the x86, Commodore, Amiga and Amstrad CPC computer platforms. Emulation is done by using existing (open source) emulators which are carefully selected on their capability to mimic the functionality of these platforms.

More information about the EF can be found at: http://emuframework.sf.net

1.4 About the KEEP project

KEEP (Keeping Emulation Environments Portable) is an international research project co-funded by the European Union 7th Framework Programme. It does research into an emulation-based preservation strategy and develops several tools to support that. The consortium consists of eight organisations representing a wide range of stakeholders in Europe: cultural heritage institutes, research institutes, commercial ICT partners and the gaming industry. The project has a duration of three years and ended February 2012.

More information can be found on the KEEP website: http://www.keep-project.eu

1.5 Terms & definitions

The term *remote desktop* refers to software or an operating system feature allowing applications, often including graphical applications, to be run remotely on a server, while being displayed locally (at the client machine). A main use of remote desktop software is remote administration; it can also be used for *headless computers* or *thin clients*.

In the context of the EF, its use is slightly different since a single application running on the server needs to be tunnelled to a specific user on a client machine giving him/her complete control over the application (in this case an emulator). *Remote emulation* thus means: to run and control emulators over a network where the emulator runs on a server and the interaction with the emulator takes place on a client machine.

2 Requirements

To make a valid comparison between existing remote emulation solutions, this section describes briefly some business and technical requirements.

2.1 Business Requirements

Cost-efficient: a lot of solutions are expensive; due to per user licence fee can others can get very expensive. Since the EF is an open-source project, the solution, if not free, will have at least to be cost efficient for the organisation.

2.2 Technical Requirements

Platform independence: the EF has been developed in Java to allow cross-platform deployment. Similarly, the remote emulation solution will have to be deployable on multiple platforms and also act cross-platform. For example, the EF core functionality could be deployed on one operating system whereas the EF client could be deployed on a different operating system.

Scalability: potentially the EF could be deployed on a server being accessed by dozen of users at the same time (e.g. a reading room in an archive or library), so the solution needs to scale well with increasing number of users.

Performance: the emulators being per definition virtualization technologies, adding many layers of virtualizations might slow down the application and render it almost unusable.

Security: some technologies do not use encryption so that the data sent over the network is not encrypted. Some solution also display to the user the entire server screen instead of a single application, making it a security breach since any user would be able to start administration tools.

Session mode: on top of being a security breach, sharing the entire server screen will make the EF unusable if several users start the same or different emulators at the same time. The user will then effectively see all the emulators on the server screen causing windows overlapping issues.

3 Technical Analysis

Remote emulation requires a protocol for connecting the client interface to the remotely running emulation service on the server. Rather than developing a new protocol from scratch existing remote computing protocols can be used.

Remote desktop applications typically use either the *Remote Desktop Protocol (RDP)* or *Virtual Network Computing (VNC)* protocol. In order to establish a remote connection, both the host/server and client have to support the same protocol. The problem is that not all operating systems use the same default protocol making it difficult to find cross-platform solutions.

Remote desktop applications have varying features. Some allow attaching to an existing user's session (i.e., a running desktop) and "remote controlling", either displaying the remote control session or blanking the screen. Taking over a desktop remotely is a form of remote administration.

Besides RDP and VNC there are other remote access solutions such as Citrix XenApp and Oracle Secure Global Desktop which will be discussed in this study as well.

3.1 Virtual Network Computing

Virtual Network Computing (VNC)[1] is a graphical desktop sharing system that uses the RFB protocol to remotely control another 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.

VNC is platform-independent – a VNC viewer on one operating system may connect to a VNC server on the same or any other operating system. There are clients and servers for many GUI-based operating systems and for Java. Multiple clients may connect to a VNC server at the same time but all clients will see the same screen since VNC simply displays whatever happened to be on the server's screen.

Most VNC applications send the whole desktop, rather than a single application. Exception seems to be UltraVNC[2] which handles sessions but only for the Windows version and X11VNC[3] which does the same for Linux.

However these solutions are not capable of differentiating between two users running a copy of the same emulator at the same time (two versions of the emulator effectively running on the server at the same time). Furthermore, audio support is missing in the protocol.

There are a number of variants of VNC which offer their own particular functionality; e.g., some optimised for Microsoft Windows, or offering file transfer (not part of VNC proper), etc. Many are compatible (without their added features) with VNC proper in the sense that a viewer of one flavour can connect with a server of another; others are based on VNC code but not compatible with standard VNC.

In its simplest form, the VNC protocol can use a lot of bandwidth, so various methods have been devised to reduce the communication overhead but in all cases, VNC will be slower than any corresponding RDP.

---

[1] VNC, available at: http://ipinfo.info/html/vnc_remote_control.php

[2] UltraVNC, available at: http://www.uvnc.com/

[3] X11VNC, available at: http://www.karlrunge.com/x11vnc/

By default, VNC is not a secure protocol. However, VNC may be tunnelled over an SSH or VPN connection which would add an extra security layer with stronger encryption. Nevertheless, even if tunnelled over an SSH connection, VNC simply shows whatever is on the screen with no forced logins required. This has security implications; if you remote into a machine that an Administrator is logged into, the user will effectively be an Administrator.

List of most popular VNC software:

- UltraVNC (free): supports the use of an open-source encryption plug-in which encrypts the entire VNC session including password authentication and data transfer. It also allows authentication to be performed based on NTLM and Active Directory user accounts. However, use of such encryption plug-in makes it incompatible with other VNC programs.
- TightVNC[4] (free): implements a video hook driver. One of the long-running historical weaknesses of the VNC protocol was that it didn't interface at the video driver level with Windows; it had to poll for screen changes making it inefficient and highly CPU intensive. But even with video hook driver it isn't as fast as Remote Desktop.
- RealVNC[5] (free, Personal and Enterprise Edition): offers high-strength encryption as part of its commercial package
- Guacamole[6] (free): Guacamole is an HTML5 + JavaScript (AJAX) viewer for VNC, which makes use of a server-side proxy written in Java. The current version is almost as responsive as native VNC and works in any browser supporting the HTML5 canvas tag. This is a server-side alternative that lets anyone with a web browser use VNC without the need for Java or plug-in on the client side. Guacamole translates the VNC protocol into a form that JavaScript can conveniently handle: XML with PNG images.

3.2 Remote Desktop Protocol

Compared to VNC, Remote Desktop Applications using Remote Desktop Protocol (RDP) are tightly coupled to a specific platform but they are also seen as far more efficient than VNC, due to the semantic advantage.

3.2.1 On MS Windows: Remote Desktop Service

Microsoft Remote Desktop Service[7], formerly known as Terminal Services, is one of the components of Microsoft Windows (both server and client versions) that allows a user to access applications and data on a remote computer over a network, using RDP. Only the user interface of an application is presented at the client.

Terminal Services is Microsoft's implementation of thin-client terminal server computing, where Windows applications, or even the entire desktop of the computer running terminal services, are made accessible to a remote client machine.

Clients exist for most versions of Microsoft Windows (including Windows Mobile), Linux (rDesktop), Unix, Mac OS X, Android, iOS, and other modern operating systems. Microsoft's

[4] TightVNC, available at: http://www.tightvnc.com/

[5] RealVNC, available at: http://www.realvnc.com/

[6] Quacamole, available at: http://guac-dev.org/

[7] Microsoft Remote Desktop Service, available at: http://www.microsoft.com/en-us/server-cloud/windows-server/remote-desktop-services.aspx

official client software is currently referred to as Remote Desktop Connection, formerly "Terminal Services Client".

The protocol uses 128-bit encryption (RC4 encryption algorithm). RDP is also possible over a VPN connection, but it is not required for a secure session, according to Microsoft. It is possible to have audio transferred across the RDP, and even printers and devices.

The RDP is aware of controls, fonts, and other similar graphical primitives. This means that when rendering a screen across a network, this information is used to compress the data stream significantly. One such example is a region of the screen occupied by a button, with the colour grey: it is not necessary to send an image of the button across the network, but merely the information such as location of this button, size and colour.

This makes RDP much more efficient than VNC technology which is "dumb" in this respect, and largely functions by sending the actual images across the network.

RemoteApp (or TS RemoteApp) is a special mode of Remote Desktop Services, available only in Remote Desktop Connection 6.1 and above (with Windows Server 2008 being the RemoteApp server), where a remote session connects to a specific application only, rather than the entire Windows desktop.

In order to use Remote Desktop Service (RDS), a license is necessary.

The Windows Server 2008 R2 licensing model requires a server license for each running instance of the server software. In addition to a server license, a Windows Server Client Access License (CAL) is required to access the Windows Server software. To utilize the RDS functionality of the Windows Server software, a Client Access License (TS/ RDS CAL) is required as well. In Windows Server 2003 and 2008, there is a choice between Per Device and Per User CALs:

Count the total number of connecting users and the total number of clients these users connect from (not concurrent!).

- If there are more users connecting than clients (like in a classroom or multiple shift situation, where several people connect to the TS from shared workstations), then Per Device licensing is cheaper.
- If there are more clients than users (users connect from multiple clients, i.e. their office PC, a laptop and their home PC), then Per User licensing is cheaper.

3.2.2 On Linux

There are different Remote Desktop technologies for remotely connecting into a Linux server from an OS independent client. Below are a list of the most popular one but the Linux community is very active in this domain so this list might become obsolete very quickly.

NX Software/FreeNX

NX technology[8] is a computer program that handles remote X Window System connections. It wraps remote connections in SSH sessions for encryption. NX software is on par with Windows' own Remote Desktop Protocol (RDP) suite as far as performance goes. It is currently available for Linux and Solaris server. NoMachine[9] has clients available for Windows, Mac OSX, Linux and Solaris.

[8] NX technology, available at: http://en.wikipedia.org/wiki/NX_technology

[9] NoMachine, available at: http://www.nomachine.com/

Although designed primarily to optimize X11 sessions, NX server can be configured as a proxy server to tunnel Remote Desktop Protocol (for Windows Remote Desktop Services sessions) and remote Virtual Network Computing sessions (most modern general-purpose operating system platforms), giving them some of the same speed improvements. NX Software can work in session mode and display single applications instead of the entire desktop.

FreeNX[10] is the open source free implementation of NX Software.

XRDP

XRDP[11] is an RDP server that runs on Linux; it uses the remote desktop protocol to present a GUI to the user. The goal of this project is to provide a fully functional Linux terminal server, capable of accepting connections from rdesktop and Microsoft's own terminal server / remote desktop clients. Unlike Windows NT/2000/2003/2008 server, XRDP will not display a Windows desktop but an X window desktop to the user.

It is unclear if XRDP can be configured to run in session mode.

3.3 Combination of VNC server and Virtual Machine

One way to solve the VNC session mode issue is to use a system virtual machine in combination with a VNC server. A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Each time a user starts an emulation process and calls an emulator, a virtual machine containing a VNC server is created on the main server where the Emulation Framework is deployed. The emulator then runs within the VNC server in the virtual machine on the main server.

The main issue with this solution may be the performance since there are 3 layers of virtualisation: the virtual machine, the VNC server, the emulator. Furthermore when multiple VMs are concurrently running on the same physical host, each VM may exhibit a varying and unstable performance. VirtualBox[12] runs on multiple platforms and provides an API which could be used to start multiple sessions of identical VMs (using the same disk image). It is also free.

This solution has not been tested during this study and would need to be investigated further to discover if there are any limitations.

3.4 GRATE (Global Remote Access To Emulation services)

GRATE[13] is a prototype developed by the University of Freiburg as part of the European project PLANETS[14] (2006-2010). It is a demonstration of a web service offering emulation over the internet. GRATE deploys a client-server architecture using a Java-based web service. The client can be executed within every Java-enabled web-browser. On the server

[10] FreeNX, available at: http://freenx.berlios.de/

[11] XRDP, available at: http://www.xrdp.org/

[12] VirtualBox, available at: https://www.virtualbox.org/

[13] Dirk von Suchodoletz & Jeffrey van der Hoeven, *Emulation : From Digital Artefact to Remotely Rendered Environments*, IJDC, issue 3, vol 4., 2009, available at: http://www.planets-project.eu/docs/papers/SuchodoletzVanderhoevenIJDC.pdf

[14] PLANETS project, available at: http://www.planets-project.eu/

side, GRATE enables the deployment of hardware emulators like QEMU[15] or Dioscuri[16] and their remote GUI-usage within a Java-based web-browser.

The remote desktop functionality is completely based on TightVNC, resolving the session issue mentioned earlier.

3.5 Embedding the VNC server directly in the emulator

The University of Freiburg which developed GRATE has recently been working on embedding a VNC server directly in the emulator (QEMU and Dioscuri)[17].

This solution has the advantage of removing a layer of virtualisation which should thus enhance the performance of the remote emulation. However, the current state of development is unknown.

3.6 Citrix XenApp

Citrix XenApp[18] (formerly Citrix WinFrame Server, Citrix MetaFrame Server and Citrix Presentation Server) is built on top of the Windows Terminal Server platform which was originally developed by Citrix but it is more flexible than Windows Terminal server. XenApp can either host applications on central servers and allow users to interact with them remotely or stream and deliver them to user devices for local execution.

Allegedly, XenApp is available for multiple platform but Citrix could only confirm that there is a version available for Solaris (SPARC and x86), AIX, and HP-UX.

It is important to note that in addition to concurrent user Citrix licensing, there must exist a Terminal Server Client Access License (CAL) and a Windows Server CAL from Microsoft for each client connection. Both products must be adequately licensed for the environment to function correctly. There are Citrix Viewers for several platforms, including a platform independent Java Applet Viewer for Web browsers and a Java client is also available. XenApp utilizes Citrix Systems' proprietary presentation layer protocol or thin client protocol called Independent Computing Architecture (ICA).

XenApp surpasses most remote access solutions in term of performance but is in comparison is relatively expensive.

3.7 Oracle Secure Global Desktop

Oracle Secure Global Desktop (SGD)[19] software is similar to Citrix Presentation Server, but not tied to Microsoft operating systems and is considered a competitor to Citrix's products for remote application delivery. SGD claims to “provide secure access to centralized, server-hosted Windows, UNIX, mainframe, and midrange applications from a wide variety of popular client devices, including Windows PCs, Mac OS X systems, Oracle Solaris workstations, Linux PCs, thin clients, and more”.

---

[15] QEMU emulator, available at: http://wiki.qemu.org

[16] Dioscuri, available at: http://dioscuri.sourceforge.net/

[17] Embedding VNC in emulators, available at: http://openplanetsfoundation.org/blogs/2011-05-19-remote-emulation and a proof of concept at http://dioscuri.sf.net

[18] Citrix XenApp, available at: http://www.citrix.com/English/ps2/products/product.asp?contentID=186

[19] Oracle Secure Global Desktop, available at: http://www.oracle.com/us/technologies/virtualization/061996.html

If the client for Windows, Mac OS and Linux are easily accessible on the Oracle website, only versions of the server for Linux and Solaris can be found. It is yet unclear if SGD software supports a version of the SGD server for Windows. The client requires only a web browser with a Java Runtime Environment installed.

User actions like keyboard/mouse events and microphone recordings can be submitted to the server. USB devices (such as scanners) are also supported.

With the recent acquisition of Sun by Oracle, it is yet unclear how this will affect the cost and licensing terms of SGD. As of the time of writing of this document, it has been virtually impossible to find any cost information on SGD.

4 Conclusion

Many open source and proprietary solutions are available for connecting to a computer desktop or application remotely via a network. However, there is no perfect solution as almost each solution in this report has a disadvantage regarding the before mentioned business and technical requirements (e.g. cost, performance or multi-platform support). Ultimately each organisation might have to choose its own specific solution and implement it the way it fits the organisation.

For example, is only Windows or Unix Servers are supported it might choose to use platform Specific Remote Desktop Applications whereas an organisation which already has Citrix licenses might prefer to use this option.

Whatever the situation, the solution will have to be tested in-situ and its performance and scalability will have to be assessed. A Virtual Machine with VNC server, for example, might be a good affordable solution but it will need to be tested with multiple users connecting at the same time over the same network as performance might become an issue.

The table below gives a comparison chart of the different solutions discussed in this document.

Table 1: comparison overview of existing remote emulation solutions

	Cost	Performance	Cross-platform	Session mode	Security
VNC	Free	Low	Yes	Not per default	Encryption possible but depends on VNC. No session mode.
RDP	Free for Linux Not free for Windows	Good	No	Yes	Encrypted connection between server and client are
Running emulator on client	Free	Good	Yes	Yes	Problematic since the content and software are downloaded on the client
VM+VNC server	Free for VirtualBox	Unknown but could be low	Yes	Yes	Encryption possible but depends on VNC solution
GRATE	Free	Unknown but based on TightVNC	Yes	Yes	TightVNC allows encryption only for Unix client
Emulator with embedded VNC server	Free	Good but for the moment only works for Dioscuri and QEMU	Yes	Yes	Unknown
Citrix XenApp	Expensive	Very good	Not for Linux	Yes	Encrypted
SGD	Unknown	Very good	Not for Windows	Yes	Encrypted

4.1 Proof of concept

As a proof of concept, software company Tessella has successfully integrated the EF in its digital archiving service line SDB (Safety Deposit Box)[20]. SDB is a Java based web application which core element is a workflow engine.

Tessella has developed an emulation workflow which automatically starts an emulator on the server side. Using Guacamole, a server-side alternative that lets anyone with a web browser use VNC, an emulator has been displayed inside a web browser.

This solution is a proof of concept and unfortunately not (yet) viable in a production environment. Guacamole shares the server's entire desktop giving the end user complete access to the server creating an important security issue. Furthermore, this solution requires more testing to find out if it is scalable and robust. Figure 3 shows a screenshot of running an emulation service with SDB.

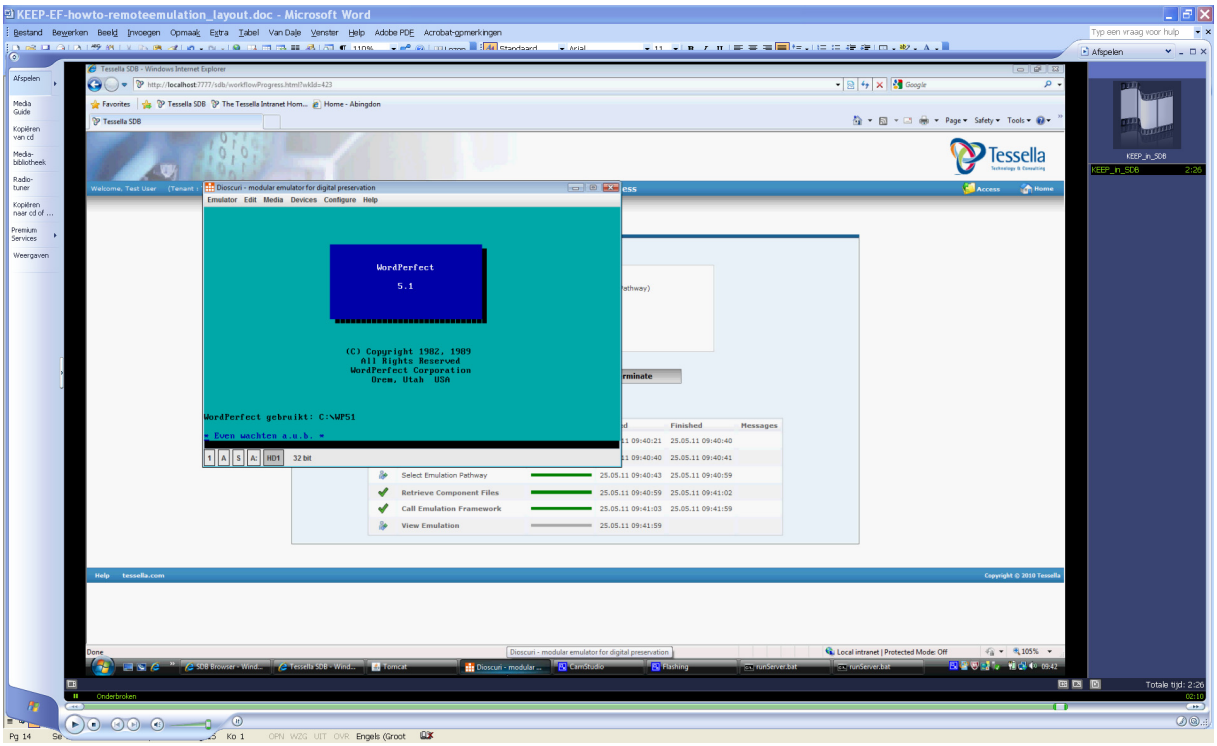

Figure 3: running an emulation service with SDB

[20] Tessella's Safety Deposit Box, available at: http://www.digital-preservation.com/

5 Questions & Answers

If you have questions regarding this document, please raise your question on the Emulation Framework support forum, available at:

http://sourceforge.net/projects/emuframework/support

Or visit the overall project website: http://emuframework.sf.net