

Ideas to Simplify a Remote Xserver Environment

Nick DeMatt
Johns Hopkins University
Applied Physics Laboratory
443-778-8092
Nick.DeMatt@jhuapl.edu

Phil Lindberg
Johns Hopkins University
Applied Physics Laboratory
443-778-3464
Phil.Lindberg@jhuapl.edu

1 Abstract

For more than a decade, engineering and information technology departments have been working on providing low cost, high quality, reliable solutions for combining engineering X-windows applications with office productivity applications. The two primary drivers for doing so are the relatively high cost of dedicated, single-user UNIX workstations and the desire to limit each user to a single desktop system. One approach is to bring the Windows environment to the UNIX workstation. Another more common approach is to serve the X-windows applications remotely to an inexpensive desktop computer running the Windows operating system. This paper describes several methods tried at the Applied Physics Laboratory and how well they performed.

2 Motivation

As stated earlier, the ultimate motivation to implement a remote Xserver environment is cost. For the engineering department it is the replacement of potentially numerous high-priced UNIX workstations with a more cost-effective centralized server environment. In some cases this would result in reduced maintenance contracts as well. For the IT department, it means fewer systems to touch, reduced network connections and possibly fewer support staff just to mention a few. While these seemingly simple goals appear to be easily achievable, success is ultimately measured by the user's acceptance and not by mere accounting numbers alone. As a result, it is quite possible and often likely that the management team in both departments may be disillusioned by premature results.

3 History

The Demise of UNIX!!

"The reports of my death are greatly exaggerated."

From the text of a cable sent by Mark Twain from London to the press in the United States after his obituary had been mistakenly published.

The early users of design automation software applications quickly discovered a problem: Most of their tools were compiled for UNIX, yet they also needed access to Windows-based office productivity tools. The simple solution was expensive and messy: put both a PC and a workstation on the user's desktop. A two-system desktop however, poses support challenges for IT as well as budget challenges for engineering management.

While the arrival of EDA tools for the PC and the emergence of Linux as a platform to combine the differing worlds has changed the environment, oftentimes it has not eliminated the problem. The reason is simple. UNIX is still the preferred environment for many engineers, while Microsoft Windows is the de facto IT department system of choice. This may eventually change if both the coveted engineering applications are fully supported on the high end X86 platform and the IT department fully embraces LINUX.

The issues listed above forced us to investigate ways to create an environment that allows use of varied design automation tools as well as standard office productivity tools from a single desktop system. Following are several case studies that review the approaches we implemented with several vendor's products, and the results of our tests.

Case Studies

Citrix Systems, Inc.

Citrix Metaframe is a suite of products designed to allow consistent access of applications through a single access point. For purposes of this case study, a Windows-based Citrix server was used to provide access to PC tools within the UNIX environment for users who already had a UNIX system on their desktop.

This approach turns one of the usual premises in this discussion on its head: Making the workstation act like a PC instead of the PC acting like a workstation.

Considerations:

A user who is already happy in front of their workstation has a lot to like about this approach. User acceptance is high because fundamentally, nothing changes for the user; capabilities are added within the current environment. They still sit in front of the same system and their UNIX application work is unchanged. But now they have access to a new window that looks and acts like a PC screen, and gives most of the functionality of a PC.

Infrastructure:

The Citrix software runs on a Windows server-class system; typical systems available during the test period handled from 10 to 20 users at a time. Our testing revealed that there were several Citrix-specific limitations on the hardware and software configuration of the system. Our IT support staff wasn't entirely happy with the system; since it looked and felt slightly different from servers they already had, they viewed it as requiring extra, specialized maintenance, and unique knowledge.

A single, centralized resource can also have the unexpected effect of delaying hardware upgrades. Since it's generally fairly expensive server hardware, replacement is often pushed off. As a single shared resource, all user performance will degrade over time. For example, in this situation, you can't replace just a single PC to solve one user's performance issues while leaving the rest of the network alone. The single access point concept central to Citrix products has advantages, but its implications must be understood.

The single, centralized server can also pose challenges when scaling this solution. Our testing uncovered services that worked smoothly on a single Citrix server

used by a small group that didn't work as well when trying to expand to multiple servers in use by a larger organization.

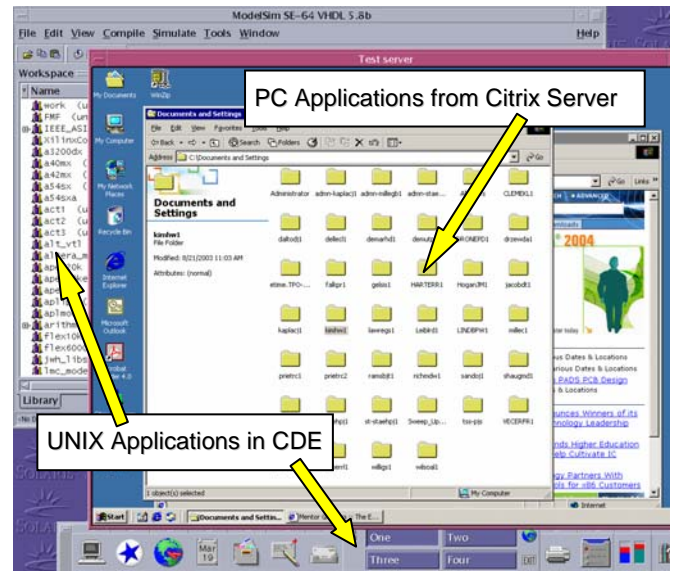


Figure 1 – Citrix MetaFrame – Sample Screen Shot

Applications:

Whatever PC applications that users need to access must be loaded on the Citrix server. They are installed in such a manner that any Citrix user has access to all applications available on the server. This can lead to some interesting situations. A user who wants to download an application off of the internet to test a new capability now has to work through the IT department to load the application on the server. The Citrix server doesn't support the concept of a user installing an application local to their workspace. While forcing all applications to be loaded in a central, controlled manner has advantages, it can also be viewed as a constraint by users accustomed to managing their own local applications.

Not being able to load applications on a per user basis may lead to other problems as well. Vendors who license products based on the number of system installations may require purchases of additional licenses or even a site license, since a single installation on a Citrix server can result in multiple users.

Storage Management:

The Citrix server fits well into an existing PC-based file system management and works seamlessly with existing PC file servers. The challenge is getting the UNIX and PC file systems to work together nicely. The

Citrix server doesn't really address this problem, as when a user expresses frustration that point-n-click methods won't work to attach a UNIX file to a PC generated e-mail message running through the Citrix server.

User Interaction: The Citrix server will look just like a full-function PC on a UNIX workstation – most of the time. For example, cutting and pasting text between the two environments works well. Moving graphics between the two environments is a different story. If a user wants a screen shot of a UNIX application to appear in a document being composed in the PC environment, some specialized knowledge of graphic formats and translation will likely be required.

There's also the matter of portable media. Widely available floppies, CDs and key-chain drives are very convenient for basic file transfer outside of the local network. However, the CD drive, floppy drive, or USB port on the local UNIX workstation could not be seen from the Citrix PC desktop window. The A: drive in the Citrix window is still the floppy drive – on the Citrix server itself, not the system in front of the user.

The Citrix server solution was a useful way to bring basic PC functionality to the UNIX user desktop. Ultimately, we encountered too many limitations to develop it as an enterprise-wide approach to our design automation environment.

Note that several of the limitations mentioned above are specific to the Citrix implementation we used, and the versions available at that time. Some of these limitations may have been lifted or addressed in other Citrix products, or similar competitor products.

XWindows on Windows

Since our experience with serving the Windows desktop to the UNIX workstation was less than acceptable, we turned our attention towards bringing the UNIX desktop to the PC. Two popular products available at the time were SCO's XVision (now part of the Tarantella Vision2K product family) and Hummingbird's Exceed. Since both of these products were already in use at our site, we elected to evaluate each of them.

As expected, both Exceed and XVision are similar in nature. They are relatively easy to install and use, but come with an exhaustive list of configuration settings. To our fortune, we were able to use almost all of the default settings except those related to fonts. That is

because to properly configure this type of environment, several font issues need to be addressed.

- Font collection
- Font conversion
- Font serving

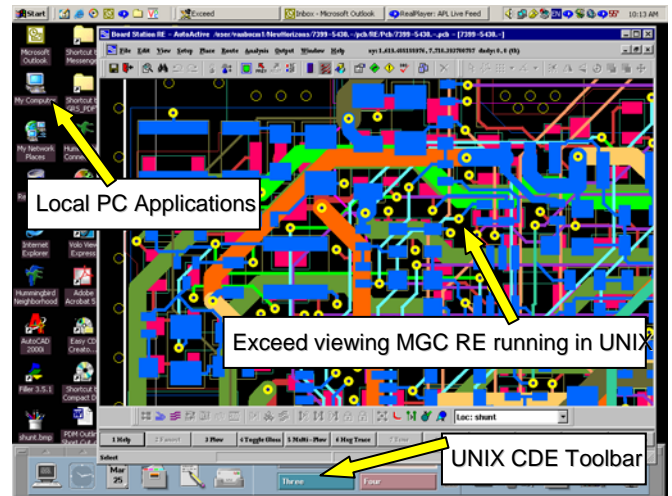


Figure 2 – Hummingbird Exceed – Sample Screen Shot

Let's take each of these one at a time. Font collection is the process of gathering all of the application-specific fonts that the X server, running on the PC, will need to properly display the application. Mentor Graphics conveniently provides a font collection utility in the `$MGC_HOME/bin` directory.

Font conversion is the process of converting the collected fonts into the format required by the Xserver. Mentor Graphics also provides a font conversion utility located in `$MGC_HOME/bin`. Just be careful to select the appropriate output format as there are several to choose from.

The third issue relates to the location of the converted fonts. There are two basic choices. Either store them local to the PC, or access them through a font server. The main advantage for storing the fonts locally is performance. However one disadvantage is that they can quickly become outdated. For instance, if a newer version of an Xwindows application is installed that contains new fonts, the whole process has to be repeated. That is the fonts must be collected, converted and added to the local directory. Likewise fonts must be collected, converted and loaded on the PC for each new application as they become available. But the greatest disadvantage is the effort associated

with managing distributed font directories to a large number of users in concert with replacement PCs. These issues are problematic for both users and computer system administrators.

An alternative approach is to create a central font server and have the PC community point to it. Once the PC is properly configured, the user will automatically receive instant and continuous upgrades to the font database as soon as they occur. In this configuration, both the user and the administrator are very satisfied.

Even this approach can be improved yet again by simply configuring the UNIX Xserver to include the font server path. By doing so, the PC Xserver application does not need to be told where the fonts are and therefore can be installed without changing any default configuration settings. Now the administrator has an out-of-the-box installation, thereby eliminating any annoying issues related to settings when doing future upgrades or new installs.

While the Exceed and XVision software products do provide a good working solution, there is still a nominal purchase price and a significant installation effort required for every PC. Therefore we decided to turn to the public domain arena to search for a solution. What we found was simply wonderful.

VNC – Open Source shows the way

This case study addresses the open source program called VNC (Virtual Network Computing).

VNC is a remote display system, originally developed by AT&T Labs in Cambridge, which allows you to view a computing ‘desktop’ environment not only on the local machine but from anywhere on the internet and from a wide variety of architectures. VNC is now supported and developed by RealVNC, a UK company. The VNC protocol has also been implemented in other products such as TighVNC and UltraVNC.

VNC began as an open source effort and continues in that tradition. While there are commercial versions of VNC and for-charge services available, several free implementations of VNC are widely available.

For purposes of this discussion, we primarily use VNC as a remote desktop viewer for PCs accessing UNIX systems.

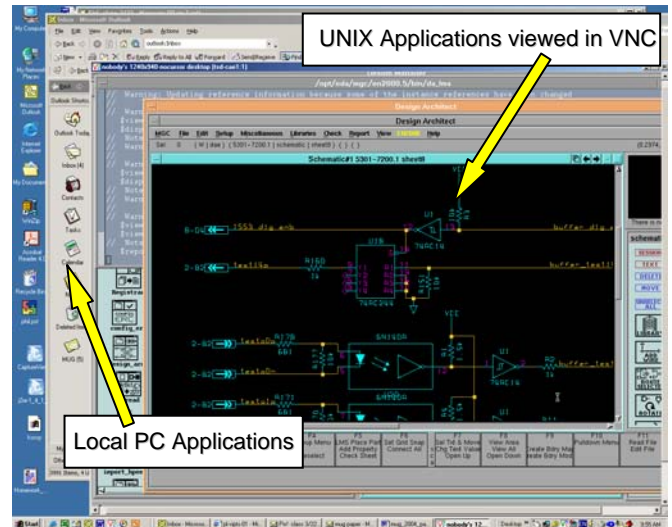


Figure 3 – VNC – Sample Screen Shot

Cost:

It's free. It's hard to argue with a product that works well and doesn't cost anything. We have used free software that is not worth what we paid for it. VNC isn't in that category.

Application Considerations:

VNC is a thin client. It has very few requirements on the viewer system and the viewer itself is under 250Kb. For example, in environments with restrictive user-installed software policies, VNC can be run merely from a copy of the executable on a Windows PC desktop. There's even a Java implementation of VNC that can allow a full remote desktop view from within a web browser.

Network Considerations:

The VNC protocol itself is light; it consumes little network bandwidth. For example, though not recommended, a remote PC can execute Mentor Graphics applications over a home dial-up connection.

Security:

For applications where VNC is run through a firewall, it's easy to implement a remote connection using SSH (secure shell) to support VNC. We recommend PuTTY, an open source utility that includes a graphical SSH implementation.

Fonts:

VNC operates differently than classic Xwindows applications. The Xserver, somewhat counter intuitively, runs on the local display system while the Xclient runs on the remote system; sadly, this system is often called a server. It may be helpful to think of VNC as a “server/viewer” system as opposed to a “server/client” system. The VNC server runs on the remote system (yes, this means the server runs on the server) while only the viewer runs on the local system. This means that all font issues are handled on the remote system, eliminating any font collection or serving issues. If the remote process can see its own fonts, which is typically simple to configure, the VNC server will transmit the graphic commands to the viewer without a problem.

Performance:

We use VNC for most of the Falcon Framework board flow applications. The performance for DMGR, DA, QuickSIM, Librarian, and FABLINK have been very acceptable. One of our earlier VNC versions however performed poorly when using LAYOUT. This condition has since been fixed with the latest version of VNC.

Disadvantages:

VNC has given us few opportunities to complain. There is one annoying idiosyncrasy we've discovered that Mentor Graphics users may notice: The “server/viewer” configuration mentioned earlier means that keyboard definitions created using Xwindows commands, such as “xset”, are ignored by VNC; the local system keyboard definitions apply. Many Mentor Graphics applications have features that involve holding down a function key while moving something with the mouse. This capability requires that key repeat be turned off on function keys. Unfortunately we have been unable to configure a Windows PC to selectively turn off key repeat for function keys while keeping it on for other keys. Since Mentor Graphics applications are famous for providing numerous ways to do any single task, this VNC limitation isn't a big issue. We believe there may be a solution to this problem waiting to be found by the right dedicated, motivated person. If you're interested in learning more, please contact the author.

There are two other minor disadvantages to VNC that we know of. One is the need to have a VNC password. The other is the need to start the Xsession on the server before launching the viewer. This is done by simply establishing an ssh or telnet session with the server and typing “vncserver”. Once the session is

active, you will be given an Xdisplay number like :2 to use with your viewer. Although neither one of these issues were very painful, we continued our research until we discovered the following.

Two ways to use VNC:

Current VNC users may not be aware that there are several ways to configure VNC. For purposes of this paper, we will call them “Traditional VNC” and “Easy VNC”.

“Traditional VNC” – The user starts a VNC server on the remote system by using ssh or a similar protocol, and then displays the session using the VNC viewer. Once started, the VNC server is available from any other network-accessible system. This means a user can leave their desk, go to another machine and re-connect to the existing session. This also means that if the local system fails, the server continues to run and a connection can be re-established later – without interrupting applications that were running in the session (no more lost UNIX sessions just because your PC crashed). This level of connectivity also means that more than one viewer can be attached to a VNC server at a time, which can be a powerful feature for remote support or classroom instruction situations.

A typical “Traditional VNC” user session would go something like this:

- From PC, telnet (or SSH) to UNIX system
- Type vncserver to start VNC server on UNIX system
- First time users will need to provide a VNC password
- Note the Xdisplay number assigned to this session
- Start the VNC viewer on PC using this display number
- Log into the UNIX network through the VNC viewer

When done working, the user can either log off, ending the VNC session, or merely close the viewer, leaving the session running. The session continues to exist, and can be re-attached from any other network-accessible system that has a VNC viewer.

As mentioned previously, one disadvantage of “Traditional VNC”: is that it requires a password to support re-attaching to a session. We consider this a disadvantage only because typical users are already burdened with too many username and passwords to remember and update.

“Easy VNC” – The user simply launches a VNC viewer on the local system while only supplying the name of the remote system. The remote system (which must be configured to support “Easy VNC”) automatically starts

a VNC server when it receives the request, and attaches it to the viewer.

A typical “Easy VNC” user session would go something like this:

- From PC, start VNC viewer using UNIX system name
- Log into UNIX network and do UNIX work
- Log out of UNIX environment when done

The “Easy VNC” approach doesn’t support session re-attachment or multiple session views; features that can be used with “Traditional VNC”. However, “Easy VNC” is more intuitive for new users, particularly those who are distrustful of their UNIX skills. It does not require a separate step to start a server session, nor does it require a password.

The combination of these two methods, open source pricing, and improved performance has made VNC our de facto remote Xserver environment choice. It is without hesitation that we highly recommend VNC to anyone who might be interested in improving their environment.

4 References

[1] Citrix Systems, Inc. web site:

<http://www.citrix.com>

[2] Hummingbird Ltd. Connectivity products web site:

<http://www.hummingbird.com/products/nc/index.html>

[3] Tarantella Inc. Vision2K integration products web site (Xvision):

<http://www.tarantella.com/products/vision>

[4] RealVNC corporation web site:

<http://www.realvnc.com>

[5] PuTTY web site:

<http://www.chiark.greenend.org.uk/~sgtatham/putty/>

[6] PuTTY and VNC installation guide

nick.dematt@jhuapl.edu

5 Acknowledgements

The authors would like to gratefully acknowledge Steve Gelsie of Johns Hopkins University Applied Physics Laboratory for his assistance with the UNIX administration efforts associated with each of the case studies.