Remote GUI access with VNC

SHARING

vanda, Fotolia

Once you only needed a command line to access remote computers, but today's networks sometimes require a graphical connection. The VNC protocol provides a practical cross-platform screen sharing solution. We'll take a close look at VNC and show you some leading no-cost VNC applications. **BY JAMES MOHR**

I work in a data center with hundreds of machines spread over two buildings. Usually we can perform the necessary management functions for the Linux and Unix systems using *ssh*; however, in a number of cases, a tool we need may not have a command-line interface, or the remote system may be running some version of Windows that requires access to the GUI.

Virtual Network Computing (VNC) is a popular alternative for sharing screens on heterogeneous networks. VNC behavior is slightly different from the X Windowing system. One difference is that VNC *shares* the entire desktop. A user on one machine can see the current desktop of a user on another and control the mouse and keyboard of the remote system. This feature is useful for a range of activities, such as accessing a work computer from home, working with students in a training environment, providing tech support, or annoying your son while he's playing video games.

With the X protocol, the local machine is responsible for managing the display and windows. VNC uses the Remote Frame Buffer (RFB) protocol, transmitting mouse and keyboard events from the client to the server, then sending screen updates back to the client.

The simplest method for updating the screen is to send the raw pixel data in scanline order (left to right, top to bottom). Once the initial screen is drawn, rather than refreshing the entire screen when something has changed, the VNC protocol uses a simple primitive to place a rectangle of data at a particular location. This way, only the areas that change need updating.

Because more information is sent across the network than with X11, VNC is obviously slower; however, in the tests I conducted, VNC did not cause any significant performance problems.

Even with limiting the data sent in this fashion, it can still cause bandwidth problems on slow connections with a lot of updates. Therefore, a number of different compression techniques have been developed to further reduce the amount of information being transferred.

Because it interacts directly with the frame buffer, the RFB protocol – and thus VNC – is platform independent. A Windows machine can connect to Linux just as well as it can connect to another Windows machine. VNC can essentially work with any windowing system and is therefore ideal for both Windows and X11, as well as Macintosh systems.

Getting Connected

Like the Windows remote desktop client, the VNC server stores all of the connection and session information. Therefore, you can disconnect a client while at one location, then reconnect later at a different location and start from where you left off. An X11 client typically cannot detach itself from the server and then reconnect itself (although there are some X11 apps that allow this).

The controlling application or client is often called the “viewer” since it is used to “view” the remote machine. This name has a farther-reaching implication, as you can actually configure the server to be “read only” or “view only.” That is, you can see the display, but you cannot make any changes. We have used this at work for remote monitoring of applications where we were not allowed to change anything, just report when something happened.

In a support environment, the technician can see the steps the user takes without being able to make changes. In other cases, you can configure VNC to show a single display on multiple machines, such as a training environment when the instructor is demonstrating something to a large number of students.

Keep in mind that VNC needs to update the local display every time you make a change such as moving or resizing a window. Not only does the window itself need to get redrawn, the background behind the window needs to get updated as well.

If you have a complex background, it naturally takes longer to update. Also, the higher the resolution, the more data needs to be transmitted, which thus effects the performance.

Xvnc

Some VNC versions provide not only VNC services, but X11 as well. Xvnc, the actual server application on Linux, offers X11 and VNC, supporting multiple concurrent X11 sessions on the same remote machine and includes connecting to existing X11 sessions as well as creating new, independent sessions. This means that what Xvnc is displaying on the client is not necessarily the same as what is running on the server’s display.

Like X11, VNC creates virtual displays or “screens” to each connection, but this is not the same as the virtual desktops you get with KDE. By default, VNC uses ports 5900 through 5906, with each port corresponding to one of these screens, so you end up with screens :0 to :6. Note the colon in front of the screen number, which is the same way displays are numbered with X11. Depending on the product, there is a built-in HTTP server to which you can connect using ports 5800-5806. In either case, you can use other ports, but you naturally need to make sure that both the server and client know which port to use.

What Goes Into VNC

Packages typically come with five programs, but the VNC server and viewer sometimes come in separate packages. Although the behavior is generally the same, you will find slight differences in the various VNC alternatives.

Packages typically appear as RPM and tgz-files for Linux, or as self-extracting *.exe* or *.zip* files on Windows. Source code is available for the open source versions, including Windows.

The *vncserver* script is what is typically run to start the VNC server. This file can be in various directories such as */usr/bin*, */usr/X11/bin*, or */usr/local/bin*, depending on the product and how it was installed (RPM or compiled). The script is simply a wrapper script to start the *Xvnc* program, which is the actual server. This script accepts far few options than *Xvnc* supports, but for the most part, the options *vncserver* accepts are sufficient. The client or viewer is provided by the *vncviewer* program.

Two support programs are also delivered. The *vncpasswd* program allows you to set the password for the current user. Passwords must be six characters long; the program will fail if the password is shorter. If the read-only password is too short, the primary password will be saved. Note also that, in most cases, only the first eight characters are significant and the rest are ignored.

By default, the file *$HOME/.vnc/passwd* is used, but you could specify a different file. For example, a system administrator who wants to create a password file for a specific user (i.e., to reset a password) can specify the password on the command-line, like this:

```
vncpasswd filename
```

If, for example, the user’s home directory is on a shared filesystem (i.e., SAMBA or NFS) and you would like to make sure the VNC password file is local, you can specify the *-t* option to *vncpasswd*, which will cause the application to write passwords into */tmp/$USER-vnc/passwd*.

The *vncconnect* program is used to connect a VNC server to a VNC viewer on a specific machine.

A couple of the VNC products I reviewed also include the *vncconfig* program, which provides two different functions. The first function is to display or set Xvnc parameters for a running server. The second function is to help with clipboard transfer to and from the VNC viewer.

Many Linux distributions provide a VNC server by default and it is usually configured as “Remote Administration,” or something similar, although you can use it for other things as well. In many cases, all you can do within the respective administration GUI is to activate/de-activate it and perhaps change the firewall settings.

Although the same programs exist on Windows, I found that the name for the server application differed from product to product. For example, TightVNC named the server program *WinVNC.exe*. Provided the necessary DLLs are accessible, you can start the server without having to actually install it.

Like Linux, the Windows VNC server can also be configured as a service that is started automatically when the system boots, but this usually means having to actually install the product. The products I looked at allow you to either dump the respective programs into a directory and start them from there, or you can actually install the programs, which creates menus and desktop icons.

I ran into firewall problems with a couple of the Linux distributions I tested. The ports used by VNC are typically not open by default, so check out the firewall configuration if you are having trouble connecting.

I also ran into another problem with Fedora 6, which offers a copy of RealVNC 4.1.2 that is well integrated into the system. The VNC server is set

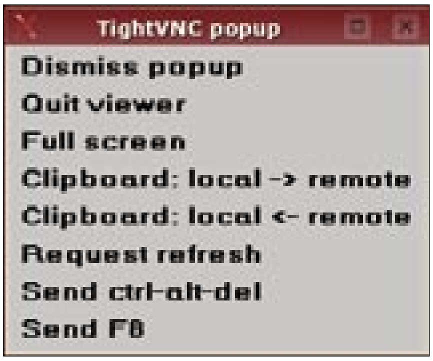
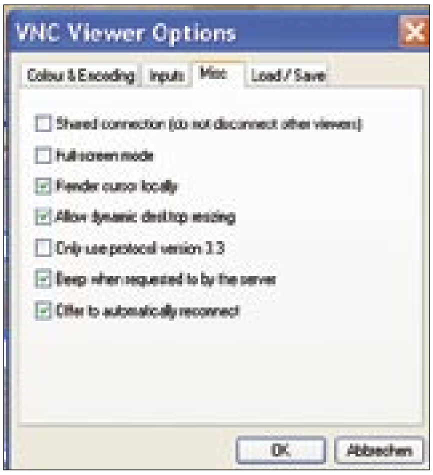


Figure 1: The Linux Viewer configuration options.



☐ Shared connection (do not disconnect other viewers)
☐ Full-screen mode
☑ Render cursor locally
☑ Allow dynamic desktop resizing
☐ Only use protocol version 3.3
☑ Beep when requested to by the server
☑ Offer to automatically reconnect

Figure 2: The Windows Viewer configuration options.

up by default as a service that you can configure and start using the service configuration application. Also, the default configuration file is stored as */etc/sysconfig/vncservers*.

When I tried to start from the admin GUI, nothing appeared to happen. From the command-line, I got the typical "Failed" message, but it wasn't until I poked around the init-script that I found that a couple of changes were necessary in the *vncservers* file to actually start VNC. The tool was also configured with the *-localhost* option, which meant that it would only allow "loopback" connections, such as when using SSH tunneling. After addressing these issues, I could connect from my local machine to the server.

Running VNC

The first step is getting the server running, which you can run as any user by simply starting the *vncserver* script. The first time you start the server, you'll be prompted to input a password for the connection.

The *vncserver* script then creates the necessary configuration files, including any applications that should start automatically, are contained in the respective *$HOME/.vnc* directory.

After prompting for the password, or if one already exists, the VNC server displays the basic information about the connection and which file it is using to start any applications. By default, applications are started by *$HOME/.vnc/xstartup*. The first time you start the browser, the viewer will typically create an xstartup file for you.

On Linux, by default, the *xstartup* script simply sets up the basic environment then starts an xterm (console) and the TWM windows manager. TWM is fairly simple by today's standards. Although I don't necessarily recommend TWM as a day-to-day desktop, it suffices for remote administration.

As you can see in Figures 1 and 2, the Windows client GUI provides many more options than the Linux GUI, even with the same product. However, as you would expect from Linux, the equivalent options are available as command-line arguments.

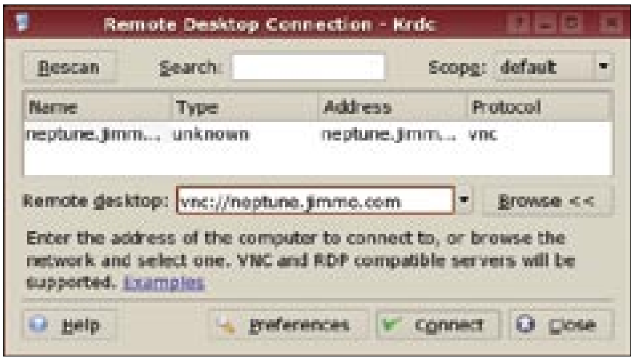
The Linux client can simply be started by running *vncviewer* from the command line. Depending on your distribution, you might have a graphical starter program for the viewer, such as KDE's KRDC (Figure 3).

If you run the viewer without any options, you are prompted to input a hostname or IP address. However, the hostname or IP address and display number can also be given as arguments.

Each of the servers I looked at come with a built-in HTTP server, which allows you to connect using a Java-compatible web browser. Although this web option is useful in principle, the speed of the screen updates was noticeably slower with all of the combinations I tried, and the quality of the graphics was not very good. Plus, I could actually watch the windows being redrawn in the web browser, whereas with the standalone viewer, the same application appeared immediately. Still, the web option allows access from places that otherwise would not support VNC.

Free VNC Products

I decided to examine a few of VNC products that were available for free. Note that, in this case, "free" does not mean just open source products. RealVNC, for example, provides a free version along with two commercial versions. Keep in mind that VNC is the protocol by which the client and server communicate, and not the program itself. Although the original VNC source code and many current products are open source, some are not.



Scope: default

Name	Type	Address	Protocol
neptune.jimm...	unknown	neptune.jimm...	vnc

Remote desktop: vnc://neptune.jimmo.com

Enter the address of the computer to connect to, or browse the network and select one. VNC and RDP compatible servers will be supported. Examples

Figure 3: The KDE Remote Desktop Client.

Documentation

Documentation is sparse and typically limited to man pages, which are about the same across the various products. Although products typically provide the same programs, their behavior may differ. Sometimes the difference is minor, but there can be a few more significant differences.

I installed the products on both Windows and different Linux distributions, and I tried a number of different combinations of servers and viewers. Although the behavior differed slightly between products, I did not run into any problems connecting the client of one product to the server of another.

I have a 100MB local network and I noticed no delays when doing something that updated a large area of the screen, such as opening up the Windows Start menu or starting a new file browser. It didn't matter whether the change was initiated on the real Windows XP machine or through the VNC viewer – there was no noticeable delay. However, when I tried World of Warcraft running on my son's Windows machine, the delays made it impossible to play.

TightVNC

TightVNC [1] calls itself an "enhanced version of VNC" and claims to include "a lot of new features, improvements, optimizations, and bug fixes," most of which require you to dig through the various change logs.

Initially, I tried to install the RPM-package for version 1.3.9. Unfortunately, I couldn't install it because the RPM-package required a newer version of the glibc (although the required version was not listed on the TightVNC download page). Then I tried to compile it myself, but unfortunately, the documentation was missing a couple of steps.

After jumping through a few hoops, I got it compiled; however, I still could not start the server correctly and my posts to the TightVNC mailing list remained unanswered.

Because the TightVNC project is hosted by SourceForge, previous versions can be downloaded from there. The next most-recent version on SourceForge is 1.2.9, but this was installed by default on my SUSE 10 system, so I did not go through the trouble of downloading it. This version worked fine. Note that the Windows version of 1.3.9 installed with no problems.

DFMirage

TightVNC provides the DFMirage mirror display driver, which runs on Windows 2000, Windows XP, and above. Like TightVNC, the driver is free and provides an even more efficient way of updating the screen information. Even without the driver, I had noticed no delays; however, after installing the driver, at least one program froze completely even without connecting any VNC client.

Once I removed the driver, the program worked fine, so there are some obvious issues with how some applications react to this driver.

One nice thing was that the server can change the desktop to a solid color rather than transmitting the wallpaper or background. You can also disable any cute XP user-interface effects. These features increase the efficiency of screen updates by reducing the information that needs to be sent.

Further increases are achieved by so-called “tight” encoding with optional JPEG compression. Per the TightVNC documentation, this encoding is optimized for “slow and medium-speed connections,” but it was unclear what connection speeds these really are.

RealVNC

Because the RealVNC company was established by developers of the original VNC, one could argue that RealVNC [2] is the heir to the original VNC legacy. RealVNC is currently available in three variants: free, personal, and enterprise.

For my tests, I used the 4.1.2 free version, which seems to be a watered-down version of the available open source products. For example, the *vncserver* script does not support a password file anywhere other than the default location *$HOME/.vnc/passwd*.

Additionally, in contrast to other products, RealVNC’s *vncpasswd* does not ask you for a view-only password. In fact, the RealVNC version does not appear to support any of the options the other products do.

Another annoyance is that RealVNC does not prompt you for a password if the password has not yet been set. RealVNC does, however, give you a hint to use the *vncpasswd* program, which will prompt your for a view-only password. The annoyance gets worse because it displays a usage message that indicates you can specify a password file or the *-t* option; however, neither is supported by the *vncserver* script. If you run

Figure 4: The TightVNC Viewer connected to Windows XP.

vncpasswd with the *-t* option and then try to start the server, it fails again, telling you to set the password.

In addition to the features supported by the free version, the personal version also supports 2048-bit RSA Server Authentication, 128-bit AES Session Encryption, and File Transfer, among other things, but it does not support Linux or other UNIX variants.

The enterprise version of RealVNC requires a separate license for Windows and Linux servers, and it also supports Mac OS X. One addition is the "Direct platform native authentication," which means you do not need to create an extra password file for each user in order to connect to the server. Instead, the tool uses the operating system's own authentication methods, thus requiring only a single password. On Windows servers, this means you can authenticate using either NT-Domain or Active Directory credentials.

On Linux/UNIX, authentication is done via NIS/NIS+. Since the system can now differentiate between users, this mechanisms also means that you can create separate environments for different users.

MetaVNC

In terms of both the scope of the product and the available information, MetaVNC [3] seemed to be the thinnest. I installed MetaVNC viewer 0.6.5, which is based on TightVNC Viewer version 1.3.9. Installing the server on Windows went fine, but I had problems with the Linux version and what little online documentation there is specifically talks about Fedora Core and not much else. However, I was eventually able to coax the packages into letting themselves be installed.

Overall, MetaVNC also seemed to be the slowest, regardless of whether it was the client or viewer (or both). Although MetaVNC to MetaVNC was still acceptable, when connecting a RealVNC or TightVNC viewer to a MetaVNC server, screen redraws were noticeably slower. This seemed somewhat odd since MetaVNC is "based" on TightVNC, unless MetaVNC adds features that actually slow things down.

Although the project was registered on SourceForge, there was almost no activity in the SourceForge forums and I found no mailing list to which I could subscribe. There is a wiki on the MetaVNC website, but it seems to be primarily of interest to developers.

MetaVNC has almost all the same features and behavior as TightVNC. There were several new options in the Windows client, but I didn't find much at all in the documentation about what these options did.

Not to be deterred by a lack of documentation, I googled a little and found a nifty little feature called the Meta Window Manager. The Meta Window Manager merges your local Windows Start menu with the Start menu of the remote machine, and you can even configure exactly how the menus are displayed. The local and remote task bars can also be mixed, plus applications on either side are interspersed when you switch between them using Alt+Tab.

UltraVNC

The UltraVNC [4] is currently only available for Windows, but it may still be worth taking a brief look at because it provides several features that other VNC products do not have.

One feature is the Mirror Video Driver, which connects the frame buffer memory directly to the VNC server, which increases the video performance of VNC and also reduces CPU usage.

Another feature of UltraVNC is support for three different types of authentication. The first type is standard VNC authentication, where the passwords are stored on the VNC server. MS-Logon I authentication can take advantage of users, domains, and groups from the Windows machine running the VNC server. MS-Logon II allows the user account to be in a different domain than the computer account (cross-domain authentication).

UltraVNC also includes an embedded Text Chat client and file transfer.

Conclusions

Despite some initial frustrations, I found that TightVNC provided the best package and most accessible information. The free version of RealVNC left too much out, and I have little or no need for any of the extras in the commercial versions (with perhaps the exception of the file transfer).

Although MetaVNC does have a few more features than TightVNC, the lack of useful documentation makes me shy away from it, particularly in a business environment. Without Linux support, UltraVNC died in the starting gate. Business environments vary, so you may need to make some different choices. ■

INFO

[1] TightVNC: *http://www.tightvnc.com/*

[2] RealVNC: *http://www.realvnc.com/*

[3] MetaVNC: *http://metavnc.sourceforge.net/*

[4] UltraVNC (Windows Server): *http://www.uvnc.com/index.html*

THE AUTHOR

James Mohr is responsible for the monitoring of several datacenters for a business solutions provider in Coburg, Germany. In addition to running the Linux Tutorial web site *http://www.linux-tutorial.info,* James is the author of several books and dozens of articles on a wide range of topics.