Lean terminal services with NX

FASTER X

NX provides fast terminal services, even over slow connections.

BY MARKUS FEILNER

High-performing GUI-based terminal services were just a dream for Linux users. Now NoMachine's NX [1] and the independent FreeNX project [2] provide the combination of tools you need for fast graphic terminal sessions in Linux. You can even connect through a browser window.

NoMachine started developing a high-performance terminal server solution in 2000. The announcement that NoMachine was releasing its own libraries under the GPL in March 2003 caused a stir, particularly because NoMachine's NX software supports major protocols like RDP (Microsoft's Remote Desktop Protocol), VNC (Virtual Network Computing), and X11. NoMachine also lets users control any KDE, Gnome, or Windows session with just a slow modem connection, which is even more surprising in light of the fact that NX supports RSA or DSA encryption.

Why NX?

NX technology lets users connect to Linux or Windows terminal servers with a Linux, Windows, or Mac client, or from almost any browser. Users can launch individual applications that the client displays seamlessly in separate windows. Linux administrators who are forced to work on Windows from time to time will be happy to see their favorite terminals, such as Gterm or Konsole, on the Windows desktop.

Local sound output, printers (see Figure 1), and storage media are included, as is access to VNC sessions via a proxy. Users are given intuitive controls for the client, and administrators can additionally access command-line tools. Session suspend and resume features are supported, and version 3 or later supports shadowing and desktop sharing. All the administrator needs to do is open port 22 on the firewall. The underlying technology relies on X Window, SSH, Rdesktop, VNC, and other free projects.

What makes the sessions much faster than conventional X forwarding with commands such as *ssh -X -C User@ < host X program* is a clever combination of caching and intelligent proxy, as well as optimization with libraries such as Zlib and JPG.

NX not only compresses all the data exchanged between the client and the server, but also everything it stores in its local cache.

Most of NX's speed relies on knowing exactly how X Window works. The poor performance that is typical of legacy X applications on narrow-bandwidth connections is mainly due to the internal structure of the standard X11 Unix desktop, which is designed for fast, high-bandwidth lines. NX uses several tricks to work around some of the performance issues associated with X.

NX Replaces X

Instead of using X11, with its many round trips, for data transmission, NoMachine's development relies on the NX protocol. The *nxserver* program communicates with the *nxclient* program. The *nxagent* on the server handles application requests, *nxclient* handles client-side rendering, and *nxproxy* compresses and optimizes.

Because both the client and the server cache X Window data, NX avoids many round trips. Under optimum conditions, the application receives a response directly from the local process, which sends the changes (deltas) to the server. On average, NX can load 60 to 80 percent of the data directly from the cache by analyzing the X protocol. Cache hit rates are often just short of 100 percent for images such as icons and pixmaps.

NX starts by checking to see whether it can respond to a request directly from

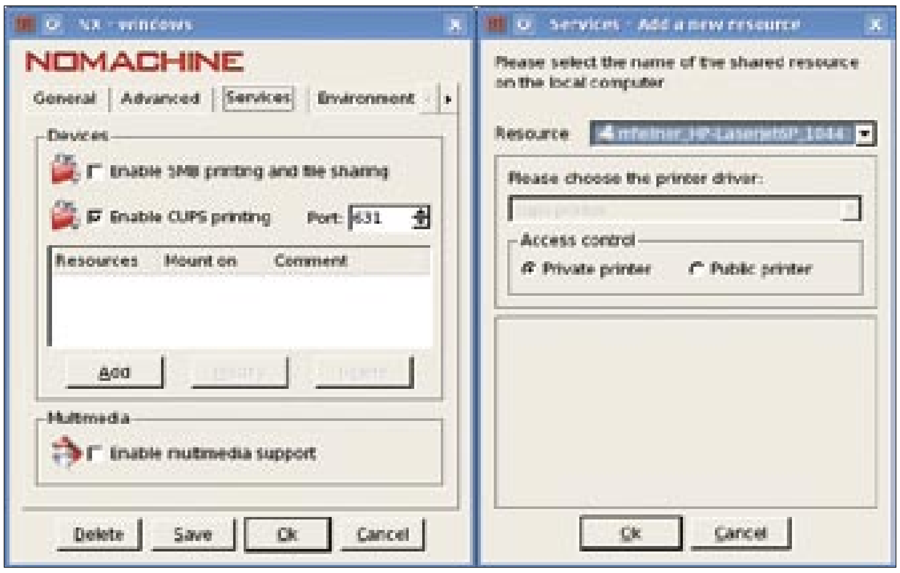


Figure 1: A local HP Laserjet is available in the terminal server session. NX relies on CUPS to bind the client-side printer to the terminal server.

its own cache; if not, *message store-based X protocol encoding* steps in. NX analyzes the X protocol and dissects its messages into two parts: a data component and an *identity*, or *fingerprint*.

The fingerprint typically occupies just a couple of bytes; the data component is the meat of the traffic. Fingerprints for two messages differ, but the probability that the data component has already been transmitted grows the longer the X session goes on. NX references and stores every single data block it transfers in its message store. Modified MD5 checksums accelerate searching. Because NX only needs to calculate the checksums to encode, and the decoding side only stores the payload, memory requirements are comparatively low. Thanks to this, the message store can hold up to 3,000 messages.

The first time a user opens an application menu, the NX client requests all of the data from the server. The server compresses the data, encrypts it, and sends the results to the client. When a second request is issued, almost all of the data is retrieved from local memory. This helps the client avoid data transmissions by fulfilling X requests from the local cache.

The message store is persistent; a user can suspend and resume a session without losing the store's content. And because NX saves the message store at the end of the session, it will be available for a second, identical session – even after you reboot the client. This method is effective, but it does give rise to security concerns. Could an attacker extract critical session data from the NX cache on a stolen or compromised laptop? This issue requires further investigation.

NX compresses the remaining data with an approach geared for optimum X protocol support and dubbed *differential X protocol encoding*. Because X messages transport vastly different content, such as images, fonts, text, and other information, and because NX speaks RDP and VNC as well as X, it uses different compression methods depending on the object and connection type: JPG, PNG, RDP, Tight, or Zlib compression algorithms. Clever combinations help the program achieve rates of between 20:1 and 2000:1.

The Zlib library still achieves a rate of 4:1 for unknown data for which a custom compression method is not available, and NX uses the same software to compress all the traffic between the client and the server. Client-side settings define a compression level between 1 (*WAN*) and 9 (*MODEM*) (see Figure 2). According to NoMachine, this last stage reduces the data volume by an additional 30 percent.

Streaming and Prioritizing

NX uses streaming in cases such as GSM (4KBps), in which the transmission of a 12KB image would take too long. The server splits the image into chunks, which it transmits at low priority after applying JPG compression. This allows the application to carry on responding to user input while NX takes its time building the background image.

Generally speaking, NoMachine gives interactive applications priority over programs that are heavy on image data by analyzing the data stream and dynamically adapting various parameters. On top of this, internal bandwidth controls, quotas, and stream compression are applied to reduce the transferred data volume. More details on NX X protocol compression are available at the NoMachine website [4].

All this effort pays dividends. Depending on the application, NX will support fairly smooth operations with a bandwidth of 10 to 20KBps and latency times in the tens of seconds. With a bandwidth of 30 to 40KBps and latencies down to 0.01 seconds, the KDE or Gnome desktop is so fast, you might think you were working locally. The multiple megabytes

dxpc

Another technology known as Differential X Protocol Compressor (dxpc [3]) works in a way that is similar to NX. The dxpc approach inspired the NoMachine developers and they made good use of analyses and tools on differential encoding of the X protocol's 160 different message types. Dxpc is still around, and according to the website, the performance achieved by the current version just about supports running an Xterm via a modem connection. It is more or less impossible to use a browser over a 28.8KBps modem line with dxpc.

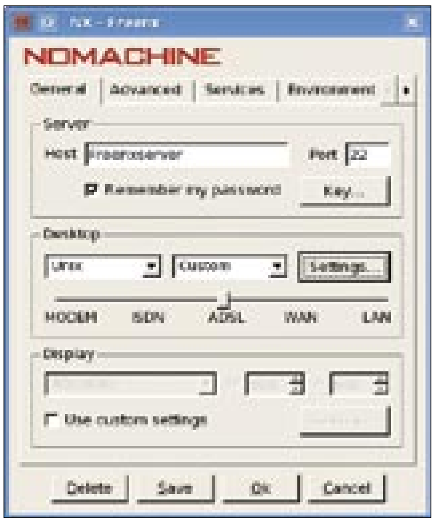
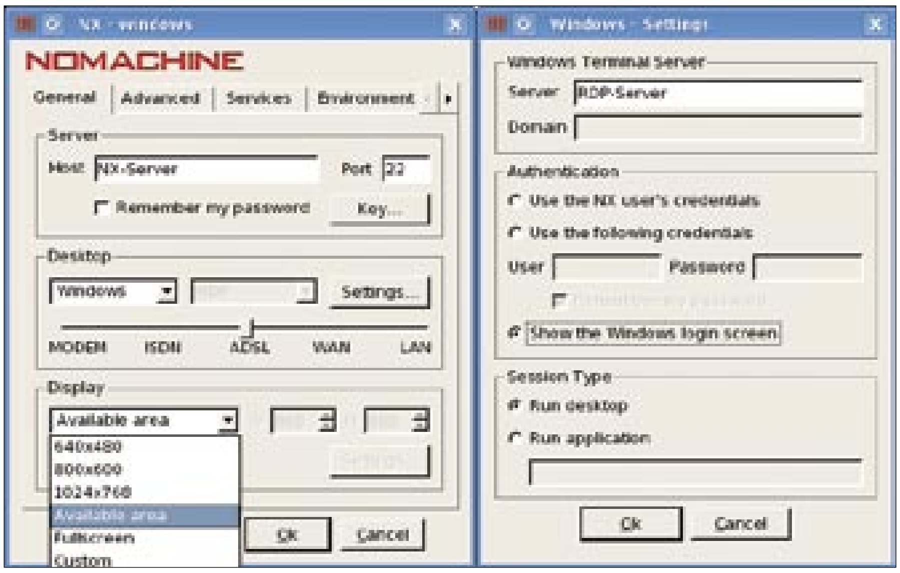


Figure 2: The slider defines the compression level. In this figure, the user has set up a custom entry to display a standalone application in seamless mode.


NX - windows

NOMACHINE

General | Advanced | Services | Environment

Server

Host: NX-Server Port: 22

Remember my password Key...

Desktop

Windows Settings...

MODEM ISDN ADSL WAN LAN

Display

Available area

640x480
800x600
1024x768
Available area
Fullscreen
Custom

Ok Cancel

Windows - Settings

Windows Terminal Server

Server: RDP-Server

Domain:

Authentication

Use the NX user's credentials

Use the following credentials

User: Password:

Remember my password

Show the Windows login screen

Session Type

Run desktop

Run application

Ok Cancel

Figure 3: Configuring the NoMachine client for access to a Windows terminal server. Users can resize Linux sessions on the fly, but it makes sense to preselect the required resolution for Windows sessions. The authentication mode is specified in the Settings dialog.

required for the first KDE start are squashed to a couple of kilobytes on next launch. This frugal use of resources means that NX will save up to 60 percent on data transfer volume over a LAN.

Clients store compressed data and decompress on the fly – with startling results in some cases. The second time you launch OpenOffice on a remote NX server, the program will be up and running far faster than on the local disk. The speeds are in stark contrast to those achieved by the overloaded, poorly configured Windows PCs typical of most office environments.

Because desktop sessions typically include frequently recurring tasks (open menu, open dialog box, move window), the longer you use NX, the better the performance becomes.

FreeNX or NX?

Several versions of NX are available, the most important of these being the Free Edition by NoMachine [5] and the packages provided by the FreeNX project. Both of these implementations are based on the same libraries, but the NoMachine free edition is newer, more stable, faster, and richer in features. NoMachine has clients for Windows, RPM, and Debian packages, source code archives, and software for Mac and Solaris.

On top of this, the developers offer a server manager and a feature called Web Companion, which lets users access Linux or Windows desktops in a browser. An embedded client gives mobile devices, such as Linux-based Ipaqs or the Zaurus, access to NX servers.

FreeNX supplies matching RPMs, Debs, and *tar.gz* archives of the current 0.7.0 version. Debian and Ubuntu users can access the repositories [6] and then call *apt-get install freenx nxclient* to install version 0.6 and Nxclient version 2.1.0-17 (Ubuntu Feisty, August 2007). OpenSUSE and Fedora still work with older versions 0.4 or 0.5.

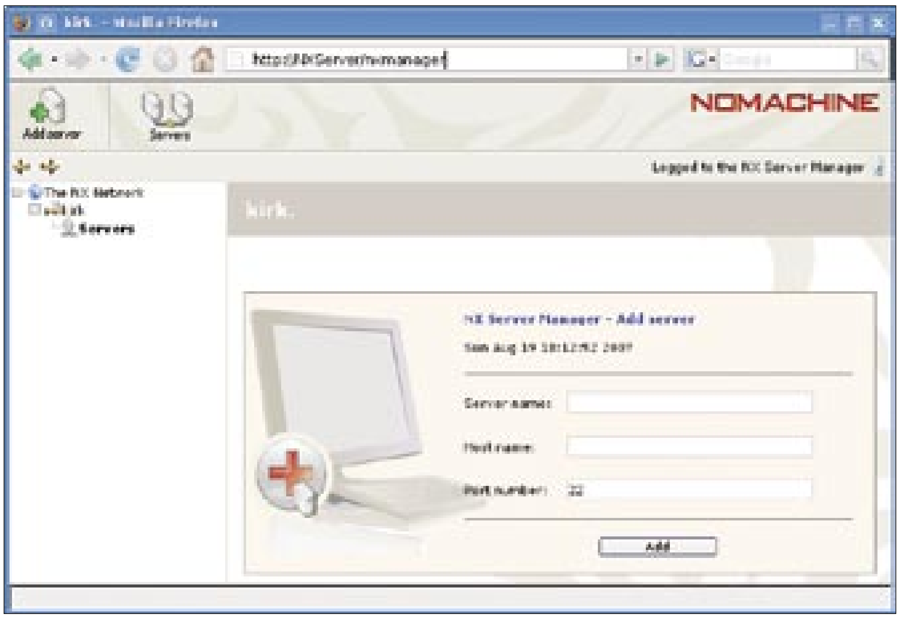
FreeNX and NX are compatible for the main part; according to the mailing list [7], both the Java applet and the Web Companion will work with FreeNX. However, the differences soon become apparent. FreeNX is much slower, and it is nowhere near as stable as the free version of the commercial product. Despite this, the Nxclient of either installation will run with servers from its competitor. Even nested NX sessions respond quickly to user interaction.

Mileage can vary on FreeNX installations, so admins can expect some manual steps. HOWTOs (e.g., for Ubuntu [8]) can help solve any issues. In contrast, NoMachine's software runs out of the box, and the client is integrated with the firmware on many thin clients.

Features and Prices

Although FreeNX is fully GPL'd, NoMachine restricts its Free edition to two users with simultaneous sessions. The commercial version supports 10 users (NX Enterprise Desktop) or 10 simultaneous sessions (Small Business Server). Enterprise Server removes these restrictions for around US$ 1,500, and it adds LDAP and Active Directory integration, a kiosk mode, and user profiles.

Advanced Server offers unlimited sessions for a price (around US$ 3,500), adding simple clustering and load-balancing to the features offered by the Enterprise Server version. NoMachine recommends SUSE, Red Hat, Mandriva, Debian, or Solaris as the operating system platform, but Xandros, Univention, and other distros are also supported.


kirk - Mozilla Firefox

http://NXServer/nxmanager/

Add server Servers

NOMACHINE

Logged to the NX Server Manager

The NX Network
kirk
Servers

kirk

NX Server Manager - Add server

Server name:

Host name:

Port number: 22

Add

Figure 4: Managing multiple NX servers in Firefox. Admins on large networks with multiple terminal servers can manage their user databases and server settings in this way.

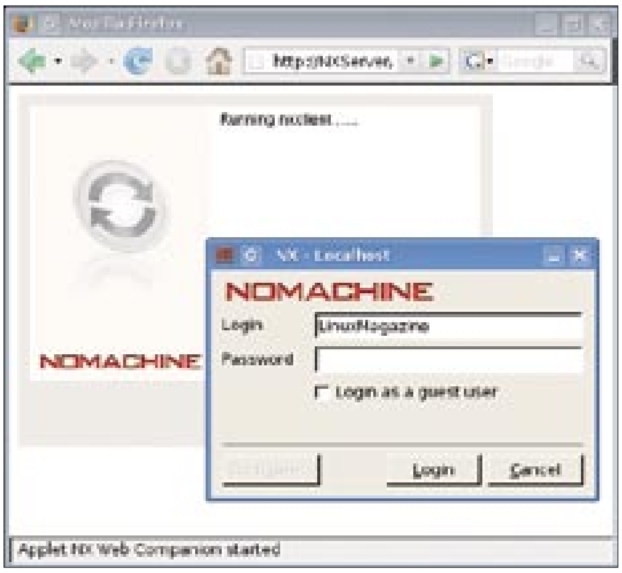
Administrators use files in */etc/nxserver* to configure FreeNX Server; the binaries are dropped into */usr/bin* and */usr/sbin* in the course of the installation, with libraries landing in */usr/lib/nx* and the documentation in */usr/share/doc/freenx*. In contrast, NoMachine does not keep to FSB recommendations and installs below */usr/NX*, which is the norm with most commercial software.

Configuration, binaries, libraries, scripts, and documentation are all available below this directory. The */usr/NX/etc/* directory contains *.cfg* files, templates, and database files for users and passwords. After the installation, NoMachine uses PAM as its authentication source, but it can also reference a user database that is completely independent of the server operating system. If needed, you can modify the setting by changing entries in */usr/NX/etc/server.cfg* (Listing 1).

The server-side options for file and desktop sharing, along with multimedia support, are also located here. The well-documented configuration files of both variants explain all the settings the server offers.

Figures 3 shows how to use the NX client for RDP-based access to a Windows terminal server. Single-application exports are possible here; you can see NX displaying single, industry-specific applications on the Linux desktop – a useful tool to speed up the migration process.

NX launches the *rdesktop* program and opens a session. It is important for the NX server – not the client – to establish the connection with the Microsoft server. The connection between the client and the NX server is again an encrypted SSH connection via port 22.



Running nxclient ....

Login: LinuxMagazine

Applet NX Web Companion started

Figure 5: Web Companion provides browser-based access to the Linux desktop. A Java applet avoids the need to install software on the client.

Server Manager (Figure 4) manages a network of NX servers (e.g., a group of servers in a load-balancing cluster). Genuine high availability is not supported, and active sessions are lost if a server dies. The Web Companion (Figure 5) tool provides browser-based access. Administrators install a standard Apache server on the NX server, follow a couple of steps detailed in the documentation, and then presto – any user with a Java-capable web browser and the right credentials is able to launch a session on the NX server.

NoMachine hides the Java plugin in the *tar.gz* version, and the RPM and Debian Web Companion packages lack the *nxapplet.jar* file. To enable browser-based access as described in the documentation, download *nxplugin-Version.tar.gz*, unpack, and store the file in */var/www/plugin/Java/nxapplet.jar*.

Conclusions

NX redefines Linux terminal services. Optimized to the max and backed with impressive know-how, NX achieves amazing performance without compromising security. Clients are available for any current operating system, and users can access both Linux and Windows terminal servers. In seamless mode, clever administrators fool their Windows users into running Linux applications like Firefox, Thunderbird, or OpenOffice – the only thing users notice is that everything is running faster. Web Companion means that a client desktop just needs a browser, and a mobile client is available, too. Trainers and support staff with NX 3 or newer can also shadow or share user desktops directly.

NoMachine shows other projects what a successful open source--based development model can look like, adding enterprise-level server products, hotlines, and support on top of GPL'd basic libraries. If you are interested in giving NX a trial run, you'll find two test servers at the NoMachine website [1]. ■

Listing 1: /usr/NX/etc/server.cfg

```
[…]
# Enable or disable NX users DB:
#
# 1: Enabled. Only users listed in NX
# users DB can login to the NX server.
#
# 0: Disabled. All the authenticated
# users can login.
# If the NX user DB is disabled, any user
# providing a valid password from
# local DB or through SSHD authentication
# can connect to the NX system.
# This is likely to be the default when
# SSHD authentication with PAM is enabled.
#
EnableUserDB = "1"
[…]
EnablePasswordDB = "1"
[…]
```

INFO

[1] NoMachine homepage: *http://www.nomachine.com*

[2] FreeNX: *http://freenx.berlios.de*

[3] Differential X protocol compressor (dxpc): *http://www.vigor.nu/dxpc*

[4] NX X protocol compression: *http://www.nomachine.com/documentation/NX-XProtocolCompression.php*

[5] NoMachine downloads: *http://www.nomachine.com/download.php*

[6] FreeNX Ubuntu repositories: *http://free.linux.hp.com/~brett/seveas/freenx*

[7] FreeNX mailing list: *https://mail.kde.org/mailman/listinfo/freenx-knx*

[8] FreeNX on Ubuntu: *https://help.ubuntu.com/community/FreeNX*