Getting Started with NX

Index

1. Getting started with NX

NX allows you to run remote X11 sessions even across slow or low-bandwidth network connections, making it possible to start sessions from clients running on Windows, Linux, Mac OS X and Solaris platforms to servers running, at present, on Linux or Solaris. Note that development for extending server support to Windows and Mac OS X platforms is in progress.

NX, thanks to exclusive X protocol compression techniques and an integrated set of proxy agents, improves the power of the X Window System to transparently run graphical desktops and applications through the network, by reducing round-trips and implementing strict flow-control of data traveling through low-bandwidth links. Even on slow or low-bandwidth network connections, you can get impressive performance thanks to NX's lazy encoding algorithm and NX's capability to automatically tune itself to network bandwidth and latency parameters.

Moreover, NX also can connect to remote RDP and VNC servers, relying on the rdesktop and TightVNC clients by encapsulating the RDP or RFB session within the X11 session.

Fig 1. A remote KDE session displayed on a local Windows Vista desktop using the NX Client for Windows.

You can identify the NX Server installed on the remote Linux machine, from the the top left of the window frame.

1.1. Some basics

The X protocol is at the base of communication between an X server and an X client. The X client is in general a program that requires a graphical interface to facilitate user interaction. The X server is instead a program that is able to draw the graphical interface of the X client and of any other running program onto the screen. The X server also handles keyboard and mouse events issued by the user and sends them back to the X client program, which then acts on the user's commands.

When the X client needs to draw something on the screen , it issues a number of requests to the X server. About 160 different types of X requests, including extensions, are specified in the X protocol. Each request represents, for example, a primitive graphic element. These requests are described, for example, at the end of the source code file named Xproto.h included in the nx-X11 Open Source component available for download **here**.

Some requests sent by the X client need a reply from the X server. Each request made by the X client, and the correspondent reply from the X server, constitute a roundtrip. Roundtrips slow down the responsiveness of a graphical program because of the time needed by the requests to complete the two-way trip.

The X protocol is network transparent. It doesn't care about where the X client and the X server physically reside. They may be on the same or different host machines. A local X server may draw on the local user's screen the graphical interface of a X client running remotely and can also send local mouse and keyboards events to the remote X client.

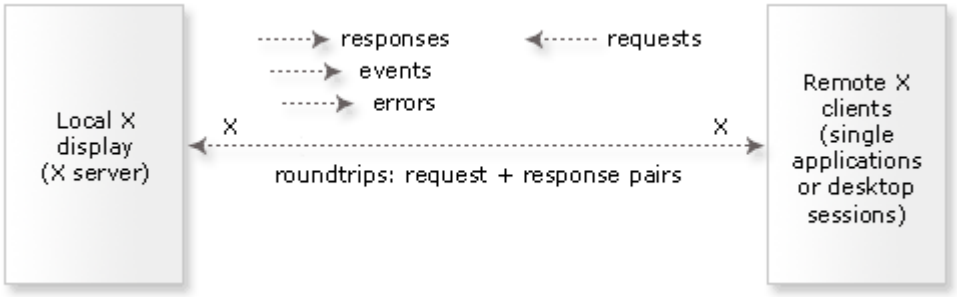
If you have a system account on a remote Linux or Solaris machine, you can verify this by running, from your local xterm:

ssh -X remote_username@remote_hostname xterm

This establishes a channel for tunneling the X11 connection from the xterm application to the local X server of the user running the command. However, the overall effect is very similar to a direct connection for the scope of this example. A new xterm should appear on your screen, if the remote SSH is configured to allow for X forwarding, prompting the remote username and host you have connected to.

Please note that when a remote application is launched and its first window is displayed on the local screen, the number of roundtrips may be near to many thousands, giving a feeling of slowness and bad responsiveness. When the X client and the X server

are running on the same host, the communication happens through UNIX domain sockets and roundtrips are reasonably fast. When, instead, the X client and the X server are running on different host machines, the interprocess communication is transported through TCP/IP network sockets and the network. In this case, roundtrips cause a slowness that is much more heavy than in the previous case.



An additional issue that can affect performance is the network latency. The quality of each link type is made up of 2 parameters: network bandwidth and its latency. Bandwidth refers to the rate of bytes that can be transferred per second while the network's latency describes how much time each packet of data needs to travel from one end to the other. Typically, a modem link has a latency of 200 to 500 milliseconds. An ADSL link has a latency of about 50 milliseconds. A local Ethernet LAN link's latency is less than 1 millisecond. A UNIX domain socket link is well below 0.1 milliseconds. You can test the latency of any network link with the ping command. The ping command shows the round-trip time in your terminal window.

1.2. Enjoy NX performance!

Using the NX method to get the best performance when the X client and the X server are running on different host machines offers:

- An efficient compression of the X traffic.
- The cache mechanism to store and re-use data transferred between server and client
- A drastic reduction of time-consuming X roundtrips, bringing their total number close to zero.

- The lazy encoding encoding algorithm to defer screen updates.

If you want to see the difference NX makes in applying these techniques, you can play with the various configurations via the NX Client GUI:

- If you select link type LAN in the NX Client GUI -> General tab, neither the NX Compression and the lazy encoding techniques will be used. Caching and round trip suppression are used instead.
- Select link type ADSL to see all these techniques applied.

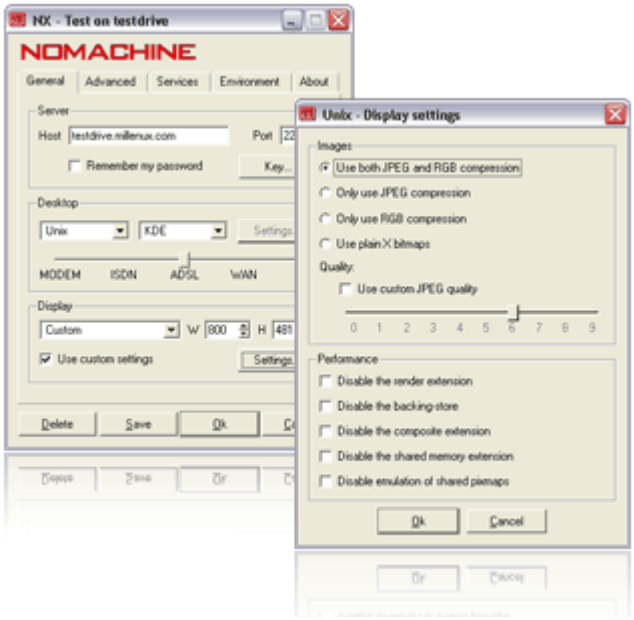


Fig.2 - The NX Client GUI: choose your settings for a KDE session to be run on the NoMachine TestDrive server, a Suse Linux 10.1 host.

1.3. Look at the NX protocol statistics

The NX Client Administrator GUI allows you to get a detailed report of the X compression statistics. By sending a USR1 signal to the NX proxy, the NX Client Administrator GUI tells it to produce total statistics, i.e. statistics starting from the beginning of the session. By sending a USR2 signal, it lets NX proxy produce partial statistics, i.e. statistics starting from the last time a USR2 signal was delivered.

To get a detailed report of X compression by means of NX proxy, you can disable

the X agent encoding. You can do this for floating window sessions by choosing 'Custom' for a 'Unix' session in the 'General' tab of NX Client GUI. Then choose 'Settings' and check the box for disabling the agent encoding.

By looking at statistics you will find that even with compression ratios in the order of 50:1, your remote session will still run very slowly. If you look at the bit-rate, running over a modem link, you will find that it rarely exceeds the 500 bytes per second. This is due to the fact most X clients use a large number of requests needing a "reply" from the X server. Such requests are marked with "R" in the NX statistics.

At any request needing a reply, NX proxy must stop compressing the traffic and wait for a response from the remote server. This is called a "round-trip". Round-trips are very expensive on links having a high latency. On a modem link, every round-trip requires nearly 200 milliseconds. If an X client requires 5 replies and NX proxy compresses these requests and replies at a rate of 1 byte each, you will get a final bit-rate of 10 bytes-per-second.

You can observe thousands of replies even running simple X applications. As you know, roundtrips degrade X performance, regardless of the ability of NX proxy to compress the traffic and regardless of the speed of your X server.

To get the best from NX X compression technology, such as the reduction of round-trips and the lazy encoding algorithm, you can evaluate the statistics with the X agent encoding enabled: the low number of replies and the overall compression ratio will highlight the NX performance against the X protocol.

2. A deeper insight of the NX technology and the NX compression

A normal X session, browsing the Internet or accessing common desktop applications, generates hundreds of megabytes of protocol data. Efficient compression is not only needed to run applications over slow-bandwidth links, but also to run multiple user sessions on common corporate LANs.

NX’s goal is to permit users to run, over the Internet, the same colourful and graphic intensive applications that run on their desktop computers. We don't try to pose any requirement on which X applications can or cannot be executed. To effectively use other remote display solutions, users have to disable fancy backgrounds, drop-down menus' animations and similar graphics effects. NX was designed to deal with such “extreme” conditions without users or desktop application developers having to modify their habits or their code. This has been our most important requirement since the beginning.

Everybody who has worked on enabling X-Window protocol to operate over low-bandwidth, high-latency links will say that the problem doesn't only reside in compression.

In fact, NX X compression is just a part (and indeed an important one) of NX proxy. It is worth noting, anyway, that many X developers consider 384 Kbps or higher ADSL connections, with latency in the order of 50ms, to be in the low-bandwidth category. In NX, when we speak about low-bandwidth, we speak about 9.6 Kbps GSM modems, with latency in the order of 500 ms. An important part of NX design and implementation was dedicated to reducing round-trips and implementing strict flow-control of data travelling through the low-bandwidth proxy link.

NX X protocol compression is derived from DXPC. The DXPC - Differential X Protocol Compressor project, released in 1995 by Brian Pane, was not only an invaluable source of ideas and a very good basis on which NX X compression was founded, it also offered specific differential encoding of many of the nearly 160 requests, replies and events that constitute the core X protocol.

2.1. The NX way of caching

Alongside the NX compression, there is the NX innovative way of caching - NX splits the X message into two parts, an identity and a data part. NX maintains a cache of the last X messages sent through the wire in the main memory, divided by protocol opcode. This cache is named MessageStore.

To allow a fast look-up of messages in the specific MessageStore, NX calculates a checksum of any new request or reply that has to be encoded. Any message type has its own method to calculate the checksum of the identity, while the checksum of the data part is simply obtained by adding any data byte to the checksum. The MessageStore's method of calculating checksum of identity has to be carefully chosen to not include those fields that are likely to change across different instances of the same X request.

When a new X request is received, NX calculates the checksum of the new message and searches it in the MessageStore. If the message is found, NX only sends this status information to the remote peer, together with the position where the message can be retrieved from store and a differential encoding of all those fields that are not part of the identity checksum. The idea of sending X updates using per-message differential algorithms was already present in the old DXPC. But the idea of splitting X messages into a 'fingerprint' and a 'data' part and caching the data part as a whole is completely new. This is what achieves the great compression boost of NX.

Over the years, NoMachine fine-tuned methods by applying a modified encoding to each of the nearly 160 X message opcodes. Now NX is in the range of 60% to 80% cache hits for the overall sample of X messages that go across the wire. For some messages, like graphic requests, images, fonts, icons and other requests used in common office automation desktop applications, cache hits can reach 100%, allowing NX to achieve

effective compression ratios in the order of 1000:1.

2.2. Roundtrips suppression and bandwidth adaptive mechanism

Think of running Mozilla 1.1 through plain X. This can require several minutes of waiting because Mozilla requires almost 6000 roundtrips to start up. With NX, start-up time decreases to less than 20 seconds, thanks to the round-trip suppression that reduces roundtrips to about one dozen.
Even on slow or low-bandwidth network connections, you can get impressive performance thanks to NX's lazy encoding algorithm and NX's capability to automatically tune itself to network bandwidth and latency parameters.

This tuning is aimed to fully exploit the network conditions by switching dynamically from an eager encoding approach (where each X client graphical request is performed on the X server) to a lazy encoding policy approach by deferring the screen updates in the case of network congestion. These algorithms keep track of areas that are out of date and determine when it is time to update them. The sensitive improvements to the responsiveness can be perceived, for example, when reproducing a video or a flash animation.

3. The NX components at work

NX is made up of a number of components that can be easily installed thanks to the packages that NoMachine offers for a great number of platforms. The communication between the local and the remote computer is achieved by setting a two-way proxying system.

Both proxies, provided by the installation of the NX Client package, communicate using the NX protocol, based on X11, but implementing extensions to compressing, caching and reducing the round-trip X traffic between the NX proxies to near-zero. In this architecture, one proxy is on your local computer. The other system resides on the NX Server.

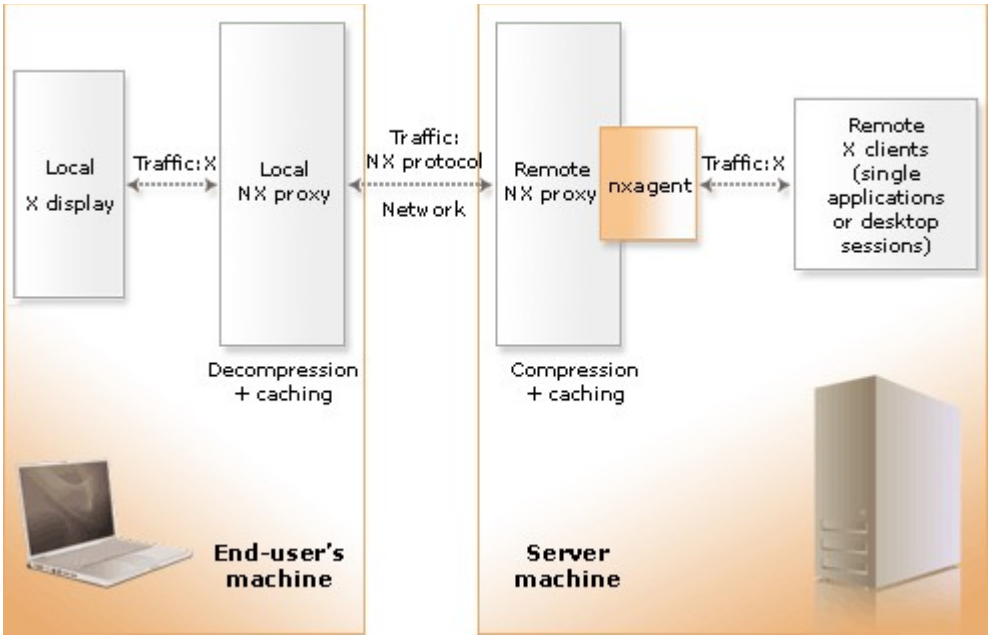


Fig 3. The NX proxying system.

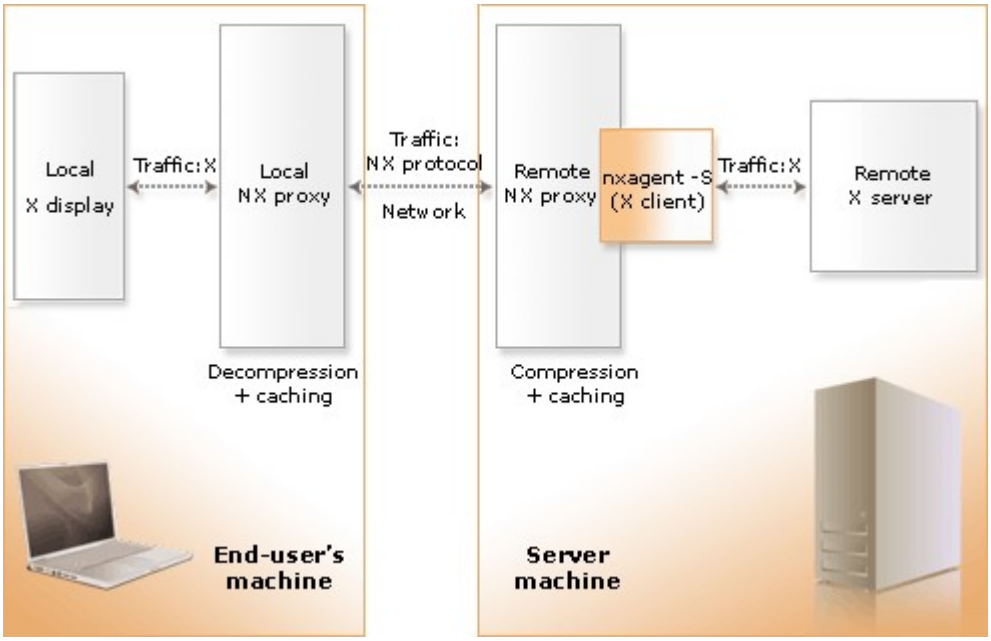


Fig 4. Desktop sharing .

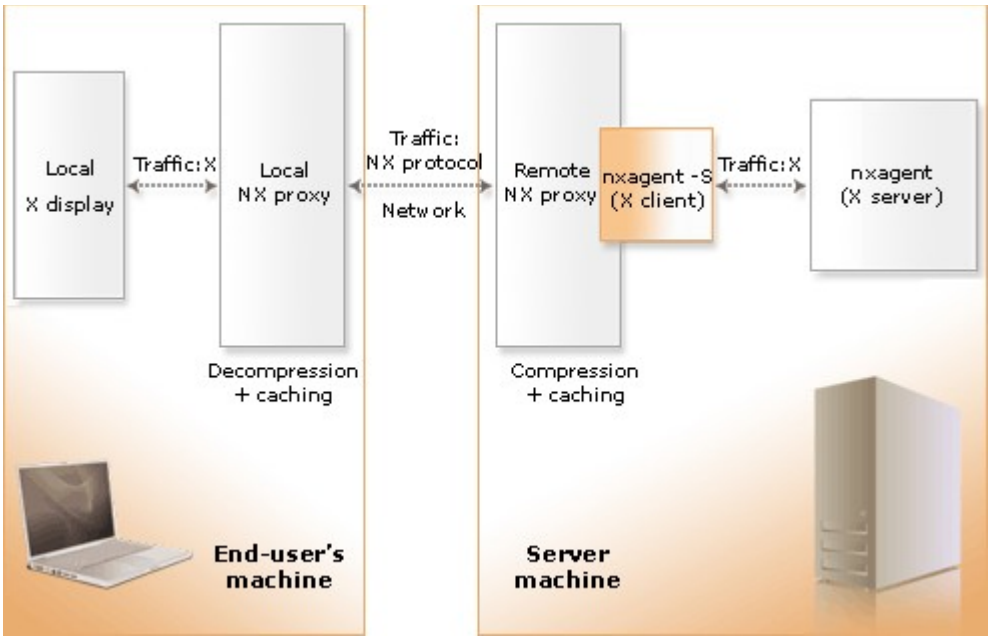


Fig 5. - Session shadowing.

The local proxy (NX Client) talks with the local X server and is in charge of translating the NX protocol back to X11. In this way, no changes are needed on the local X server to let NX work with it. The remote proxy (the NX Client installed on the NX Server side) talks to the remote X11 applications and uses the X11 protocol. Also on the remote side, no modifications are needed to the X server.

The remote NX proxy poses itself to the X clients as if it is the X server. All the roundtrips take place at this point and since they happen on the same host machine, they are quickly resolved through the UNIX domain sockets. The remote proxy, in fact, incorporates a kind of X server, named nxagent, that becomes the X server of all the remote X clients. The nxagent, also named as the X11 agent, translates the X11 protocol into NX protocol, receiving X11 requests as drawing commands and sending them to the local proxy. At this stage, the NX protocol, used for carrying compressed data or references to cached elements through the network, doesn't contain roundtrips.

Both the local and remote proxy keep their own cache of transferred data. These two sets of cache are synchronized to be useful for saving all further transmission of pixmaps, icons and so on between the remote and the local end points.

4. NX sessions insight

4.1. NX support for RDP and VNC sessions

NX supports either RDP and VNC sessions by running the RDP (rdesktop) or VNC (tightVNC) client directly within the X11 session. This provides a better level of efficiency, since the same techniques already used by nxagent to optimize network traffic are also applied in the case of Windows RDP and RFB sessions. Additional benefits also come from the possibility to suspend and reconnect these session types, as well as the possibility to access these sessions by means of session shadowing.

4.2. Full desktop sessions

NX allows you to run a full desktop session, either displayed at fullscreen on the local machine or in windowed mode, i.e. it is possible to set the preferred display size for the session window. NX supports the following desktop types: KDE, GNOME, CDE and XDM.

4.3. Single applications running in floating window mode

Single application window mode displays on your local screen a floating window drawing a single remote application, for example Konqueror or Firefox or xterm running remotely. To set-up a single application, select the desktop type Custom in the NX Client GUI and specify the application you want to run.

4.4. Session persistence

NX allows you to disconnect a session, either a desktop or a floating-window session, from the remote display, i.e the proxy agent will no longer be connected to any X client. But this doesn't have any implication on your remote applications, which will continue to be running. You will be able to reconnect the session later, even from a different machine. More information about the NX session reconnection policies are available here:

http://www.nomachine.com/ar/view.php?ar_id=AR03C00166

4.5. Desktop sharing and session shadowing

This is one of the most popular features and makes NX suitable for a full range of different usage scenarios, like remote help-desk activity and collaborative brainstorming. NX offers the possibility to connect either to a local desktop (desktop sharing of the local native display), i.e.the X server running on the NX Node host machine; or to a NX session, named the master session, running on the NX Node host machine (session shadowing). These kinds of session, named 'shadow' in both desktop sharing and session shadowing, can be requested by setting Desktop -> Shadow in the NX Client GUI-> General tab. Upon the request of the end user, the server provides to the client the list of available sessions. The user, via the NX Available sessions GUI, can then choose which session he/she would like to attach to.

4.6. Printing support and file sharing

Both the SMBFS and CIFS protocols are supported. When the session is configured to enable printers and file-sharing, NX Client specifies to the server the O.S. it is running. The default protocol is: CIFS for Windows NT-based, Linux, Mac OS X and Solaris O.S. SMBFS for Windows versions not based on NT.

5. Try a live experience on TestDrive, you are a welcome guest!

Download and install the NX Client package suitable for your O.S. from the NoMachine Web site, where you can find specific installation instructions for each package type. Let's imagine you have downloaded the NX Client for Windows. By double clicking on the .exe file, you will be guided by the Setup wizard through the installation of the client.

Then access the TestDrive page on the NoMachine Web site:

http://www.nomachine.com/testdrive

and click on any of the sessions that are at your disposal. According to the configuration of these sessions, you will be a guest user on TestDrive, i.e. NX Server will automatically generate a system account for you. Support for guest accounts is a feature offered by the NX Enterprise Server and the NX Advanced Server .

Fig.6 - Enjoy NX through the set of applications deployed on the Web thanks to the NX Builder.

All the sessions and applications in the TestDrive page are made available via the NX Builder, a php application which allows the administrator to deploy sessions on the Web.

6. The NX Web tools

NoMachine offers a set of tools which allows you to administer the NX Server and deploy sessions on the Web:

6.1. The NX Server Manager

A Web application written in PERL, using an SQLite backend. It allows you, as the administrator, to configure your NX Server(s), manage your users and sessions, monitor the status of sessions and machine availability via the NX Statistics.

Installing NX Client, NX Node and NX Server are pre-requisites to installing NX Server Manager. More detailed instructions on how to install and configure the manager

are available at:

http://www.nomachine.com/documentation/manager/install

6.2. The NX Builder

A Web application written in PHP, using an MySQL backend, created to be integrated in any Web site. It allows you to pre-configure the NX sessions you want to deploy on the Web for the end-users. Sessions files (.nxs files) will be generated on the fly and downloaded locally via the browser. NX Client is made to be the default application to run these files and as a result must be installed on each of the end-user machines. More detailed instructions on how to install and configure NX Builder are available at:

http://www.nomachine.com/documentation/builder/install.php

6.3. The NX Web Companion

Made up of a Java applet and a set of the latest available NX Client packages, it can be integrated into any Web site to allow the end-user to transparently download, install and update the NX Client locally and start pre-configured sessions directly from the Web. More detailed instructions on how to install and configure NX Web Companion are available at:

http://www.nomachine.com/documents/plugin/install

7. Set up your NX Server environment

Enter the NoMachine Web site and choose the NX Server package for your O.S.. At present only Linux and Solaris Sparc platforms are supported, but development is in progress to extend support to Windows and Mac OS X platforms. Choose the server type that better fits your needs. You can either choose between the NX Free Edition, a free-forever version of the NX Server, or any of the servers coming with a 30 day of evaluation period to let you verify which can be the best option for you.
Please refer to the following page for a complete list of features coming with the different NX Server types:

http://www.nomachine.com/features

Then open an xterm or similar on your server host and, with root privileges, install

the packages by using the RPM utility, or the graphical package manager provided by your Linux distributions. From command line, run:

 # rpm -ivh nxclient-x.y.z-w.i386.rpm nxnode-x.y.z-w.i386.rpm nxserver-x.y.z-w.i386.rpm

where x.y.z-w indicates the version of the package you are installing. You might like to verify the version of the package you have installed. You should get an output similar to the following:

 # nxclient --version

NXCLIENT - Version 3.0.0-68

Copyright (C) 2001, 2007 NoMachine.

See http://www.nomachine.com/ for more information.

 # nxnode --version

NXNODE - Version 3.0.0-76 - LFEN

 # nxserver --version

NXSERVER - Version 3.0.0-63 - LFE

Where LFE indicates that you are running the NX Free Edition for Linux and LFEN that the node is the Linux Free Edition Node.

The node and the server come with a default configuration that should be suitable for your first installation. If you want to dig deeper into the configuration files, open the /usr/NX/etc/server.cfg and /usr/NX/etc/node.cfg files. You will find a number of configuration keys, coming with a description of their purpose. Further information are available in the NX Server Administrator Guide:

http://www.nomachine.com/documentation/admin-guide

7.1. Why do I need to also install NX Client and NX node on the server machine?

The client is needed because it ships libraries used by the node. The node is needed because it ships tools needed by the server. Furthermore, a machine can be alternatively a node (in a multi-node configuration), a server or both.

7.2. Starting a session

And now your NX environment is ready for you to run your first session, but you

need to have a system account on the remote machine, since NX Server, in the default configuration, relies on system authentication. If you don't have any system account, you can create it by running in a xterm or similar, with root privileges:

 # nxserver --useradd nxtest --system

and providing a password.

Then go back to your local machine where you have already installed NX Client to take a tour on TestDrive. Launch NX Client and the login dialog will appear.

Give a name to your first session, for example my KDE, and click on the Configure button to access the NX Client GUI. You need to specify at least the server host name or IP address (this is mandatory) and the session type you want to run, let's say KDE.

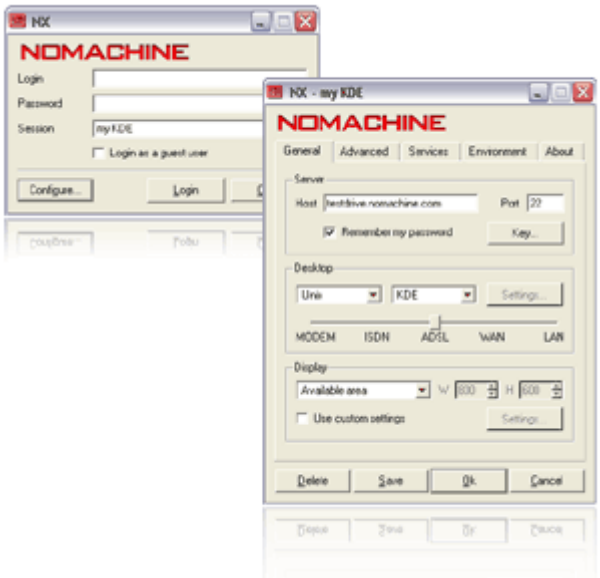


Fig.7 - Configure your first KDE session.

Then click on the Save button and come back to the login dialog, where you have to insert username and password of the system user on the remote host machine, in the sample above, nxtest.

Finally click on Login: the remote KDE desktop will be displayed on local screen.

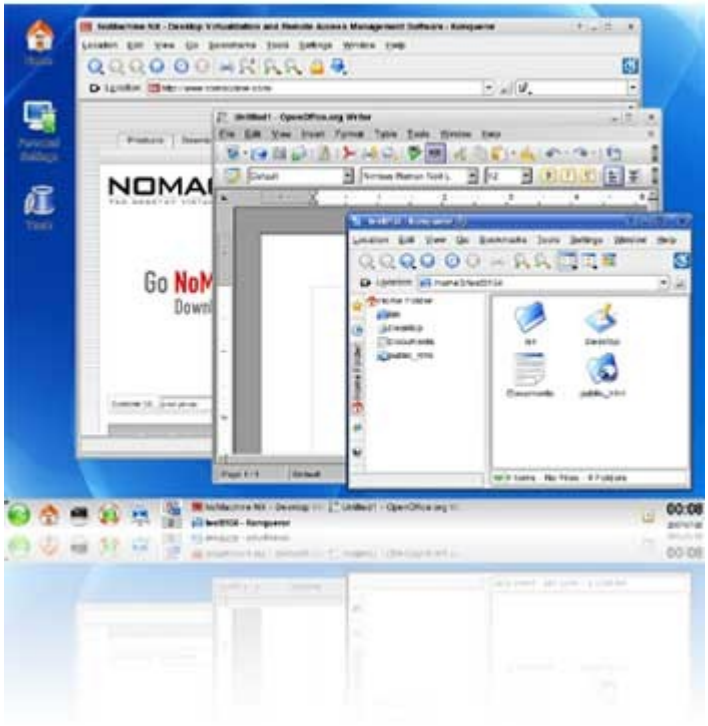


Fig.8 - A KDE session running on SuSe and displayed at fullscreen on a Windows computer.

7.3. Let's try session shadowing

If you want to experience session shadowing, i.e. the ability of attaching to a running X11 session, launch NX Client again and create a new session, named for example 'Shadowing on my KDE'. In the Configure panel, ensure that the same server host of the 'my KDE' session is specified and choose Desktop -> Shadow.

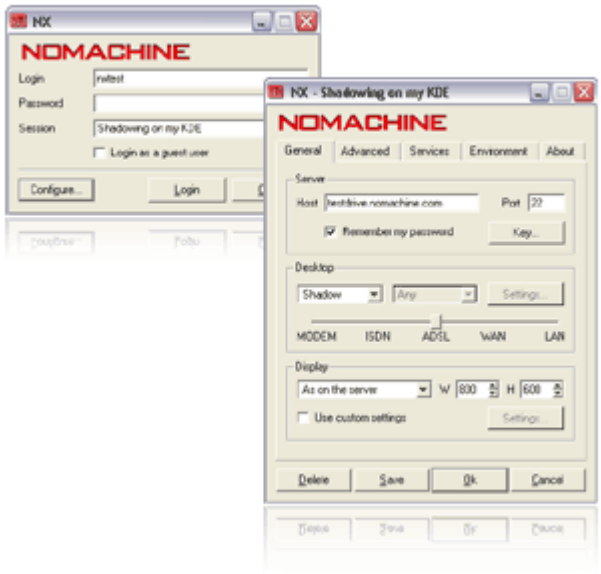


Fig. 6 - Configure your session for making session shadowing and/or desktop sharing

Click on the Save button, and keep the same credentials as before in the login dialog. Click on the login button: a list of available sessions, both NX session and local displays, is given to you. Choose to attach to the my KDE session of user nxtest. Since you are trying to shadow the session as the same user, no authorization from the owner of the master session (the my KDE session) is needed.

Now focus on the master session, the my KDE session you have started as nxtest, and click in the right upper corner to disconnect it. The session window will be closed, but all the programs you had opened inside this session will continue to run on the remote host, as you can see in the shadow session still available on your local screen.