NOMACHINE	Building and Using NX Open Source Components		
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC		
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

Building and Using
NX Open Source
Components
version 3.X

NOMACHINE	Building and Using NX Open Source Components		
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC		
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

Index

NOMACHINE	Building and Using NX Open Source Components	
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC	
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

1. Building and Using NX Open Source Components version 3.x

Follow the instructions below to build nxproxy, nxagent and the modified NX X libraries. These components are the same as those used by NoMachine's client and server to compress and deploy KDE, GNOME and Windows RDP sessions through any network connection, even extremely low-bandwidth Internet links.

1.1. Building NX Compression Libraries and Proxy

The Xcomp library and nxproxy, a stand alone program wrapping the functionalities built into the library, are the basic components you need to compress the X protocol produced by any standard X client.

In its simplest configuration, a session is run on a remote server by having a NX proxy listening to X traffic and compressing such traffic across a TCP/IP connection to a proxy peer running on the local client. X traffic is decompressed by the local proxy and forwarded to the X server.

The NX proxy makes use of "virtual display": the UNIX domain socket or the TCP port acting as a display on the application server, where the X11 protocol connection will be accepted and forwarded to the nxproxy peer running close to the X server.

Besides compression of X protocol, nxproxy is able to tunnel SMB and multimedia channels, so it is possible to listen to music from the remote terminal server or make available local resources of the thin client to the remote session.

A tunnel can be created between two hosts running:

> nxproxy -C :1000

on the remote machine, and:

> nxproxy -S remote_host:1000

on the local computer.

Most NX executables need the nxcomp and nxcompext libraries to work. Given that you built the libraries in your $HOME, run:

> export LD_LIBRARY_PATH="${HOME}/NX/nxcomp:\
> ${HOME}/NX/nxcompext:${HOME}/NX/nxcompshad:\
> ${HOME}/NX/nx-X11/exports/lib"

nxproxy is able to compress X traffic with ratios ranging from 10:1 to 1000:1 and more. The way Xcomp/nxproxy achieves these results is through extensive caching and

NOMACHINE	Building and Using NX Open Source Components		
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC		
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

differential compression of X protocol requests and replies. Differential compression means that any X protocol message is analyzed to find a similar message encoded in the past. If similarities are found, only the differences are sent through the link.

Images are encoded and compressed through a set of different methods. Encoding is usually performed by the X11 agent (nxagent), using high-level functions built in Xcompext library. The X11 agent can choose different methods, according to the type of session being run: JPEG, PNG, raw bitmaps, Z compression, run-length encoding.

Images encoded in one of the available methods are further compressed and cached by nxproxy. Cache populated at run time by local and remote proxies is saved on disk at the time session is shut down. When a new session is negotiated, a list of available cache references is sent through the link. If a matching cache is found, proxies load the cache from the disk, so further increasing compression efficiency.

A normal KDE session start-up, running through a modem link, can require as little as 35KB of data to be completed, resulting in a compression ratio of 200:1.

nxproxy is a "specialized" compressor. By "understanding" the X protocol it is able to perform many optimizations on the produced X traffic. It adapts bandwidth consumption according to network conditions and, by only compressing what is not cached by the remote proxy, it reduces the compression overhead to a minimum. By comparison, nxproxy is able to consume 1/10th of the CPU used by ZLIB compression built in SSH, while still compressing better by a factor of 10.

To build nxproxy you need:

nxcomp-X.Y.Z-N.tar.gz

nxproxy-X.Y.Z-N.tar.gz

As for all the other Open Source components, get sources from:

http://www.nomachine.com/dev_sources.php

and follow instructions you find in the README file.

> cd ~/NX

> tar zxvf nxcomp-X.Y.Z-N.tar.gz

> cd nxcomp

> ./configure

> make

NOMACHINE	Building and Using NX Open Source Components	
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC	
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

>

> cd ~/NX

> tar zxvf nxproxy-X.Y.Z-N.tar.gz

> cd nxproxy

> ./configure

> make

1.2. Running NX Proxy

In the package nxscripts there are some bash scripts made available by NoMachine to let users run NX components in sample configurations, in local-to-local conditions. Scripts can be modified and run as part of SSH remote shell procedures, to let any Linux or Unix host become a mini NX server.

We suggest you put all sources in a NX directory in your home directory. This is the default configuration. Scripts will set the required library paths according to this directory layout.

Scripts are used by NX core developers to test their software in different configurations. They should work without any modification. We suggest you try first to run sessions in local-to-local conditions, using these scripts. You can later modify them and move the remote proxy to a different host.

Let's start our first NX tunnel:

> tar zxvf nxscripts-X.Y.Z-N.tar.gz

> cd nxscripts

> ./run-nxproxy

This will run nxproxy listening on DISPLAY port 8. Scripts come with shell echo option -x disabled. If something goes wrong, you can enable this feature, just uncomment the '#set -x' line as follows:

> set -x

By simply using nxproxy you can transport any X application.

> export DISPLAY=:8

> xterm &

NOMACHINE	Building and Using NX Open Source Components		
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC		
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

> mozilla &

> gnome-session &

> startkde &

Scripts will create C-n/S-n session directories in .nx directory in your home. The session directories are used to store logs, statistics and other relevant files. An initial 'C' means that directory is storing X-client-proxy files. 'S' means that the directory is storing X-server-proxy files. Letter 'n' is the display number. In the previous case it was 8, so you should find a C-8 and a S-8 directory in your .nx.

Naming follows the X convention. The Client is usually remote, where X Clients are running. The Server is local, where the X server is running.

Persistent cache is created in the directory .nx/cache. Multiple cache directories can exist, according to an arbitrary session type, provided by the user or NX Server at session start-up.

For nxproxy configuration there are many parameters available. Given that the nxproxy tunnel is presently created as part of the agent start-up, the parameters are intended to be passed on the nx display (f.e. nx/nx,link=wan,cache=4M), not on the command line. Use of the '-' switches in nxproxy, except -S and -C, is strongly deprecated. All the available parameters are described with detailed comments, in the ParseEnvironmentOptions() code in nxcomp.

Recent versions of NX pass the parameters in an 'options' file, into the .nx/C|S-session_id directory created at session start-up. nxagent gets nxproxy parameters using an X11 protocol pseudo-extension (an extension that, for now, is only targeted at communication with the underlying NX proxy system and is not part of the X11 standard). The interface is described in the nxcompext library and it is only available to applications. The nxagent component (as well as the other agents) adapts its internal operations according to parameters reported by NXGetControl- Parameters(). For example, to get PNG image encoding instead of JPEG, you can run nxagent as in the following example:

> export DISPLAY=nx/nx,link=modem,pack=16m-png:8

> nxagent :8

Or:

> cd $HOME

> mkdir .nx/C-8

> cat >.nx/C-8/options

NOMACHINE	Building and Using NX Open Source Components	
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC	
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

link=modem,pack=16m-png:8^D

> export DISPLAY=nx/nx,options=/home/test/.nx/\

  C-boezio-1068-A2360A325FA58AC249B9F893B20D7BED/options:1068

> nxagent :8

Options are better passed in an options file as the parameters can change along the life of the session, such as when a session is reconnected at a later time on a different computer.

1.3 Getting X Protocol Statistics

To get a detailed report of X compression statistics do the following:

> cd ~/.nx/S-8

> ps auxw | grep nxproxy

lele 8899 0.6 1.3 7645 6824 ? S 14:18 2:32 nxproxy -S ...

> kill -USR2 8899

> more ~/.nx/S-8/stats

Sending a USR1 signal we tell nxproxy to produce total statistics, i.e. statistics starting from the beginning of the session. By sending a USR2 signal, we let nxproxy produce partial statistics, i.e. statistics starting from the last time a USR2 signal was delivered.

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 that 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, nxproxy 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 nxproxy 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 nxproxy to compress

NOMACHINE	Building and Using NX Open Source Components	
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC	
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

the traffic and regardless of the speed of your X server.

1.4 Building NX X Libraries and the X Session Agent

To get the best from NX X compression technology, you will need nxagent. This is a virtual X server, running on the X applications side, used to embed and deploy X desktops.

To compile nxagent you need:

nxcomp-X.Y.Z-N.tar.gz
nxcompext-X.Y.Z-N.tar.gz
nxcompshad-X.Y.Z.tar.gz
nxproxy-X.Y.Z-N.tar.gz
nx-X11-X.Y.Z-N.tar.gz
nxauth-X.Y.Z-N.tar.gz
nxagent-X.Y.Z-N.tar.gz

Go to your ~/NX directory and untar all the previous files. The nx-X11 component is a modified X.Org distribution, so, as usual:

> make World

This will also build nxcomp, nxcompext, nxcompshad and all the required libs in nx-X11.

nxagent is usually run by leveraging the Xcomp capability to provide transparent X compressed transport. nx-X11 distribution provides a new Xlib X transport layer with new X display semantics.

In order to allow the connection to the remote display, you have to add a cookie for the agent display:

> xauth add :1000 MIT-MAGIC-COOKIE-1 a0c6dcc4525d8616f9cc6098ca985f34

> xauth add localhost:1000 MIT-MAGIC-COOKIE-1 a0c6dcc4525d8616f9cc6098ca985f34

You can run:

> ./nxagent -display nx/nx,link=modem:1000 :1000

The modified Xlib, linked by nxagent, will create a proxy process that will listen for connections from a proxy peer running on the user's client. In this way any X client can be "NX-ized". The local (thin client) proxy is run with:

> ./nxproxy -S nx/nx,cookie=a0c6dcc4525d8616f9cc6098ca985f34,connect=remotehost:1000

NOMACHINE	Building and Using NX Open Source Components	
Prepared by: Marcello Blancasio	*N°:* D-811_003-NXB-OSC	
Approved by: Silvia Regis	*Signature:*	*Date:* 12/11/2008	*Amended:* A

The local nxproxy will connect to the nxagent process and forward the X traffic to the current display. The string specified in the option 'cookie' acts as a fake cookie: the X server side proxy will forward the authorization credentials by replacing the fake cookie with the real cookie, as it is read from the auth file using the xauth utility.

nxagent is based on the well known Xnest "nested" server. nxagent, like Xnest, is an X server for its own clients, and at the same time, an X client for the real X server.

The main scope of nxagent is to eliminate X round-trips or transform them into asynchronous replies. nxproxy does not make any effort to minimize round-trips by itself, this is demanded from nxagent. Being an X server, nxagent is able to resolve locally all the property/atoms-related requests, ensuring that the most common source of round-trips is nearly reduced to zero.

As we have seen, good compression would be useless without a comparable effort dedicated to reducing the X round-trips. Single applications run through an nxproxy tunnel can leverage round-trip suppression by means of a rootless nxagent, i.e an nxagent that is able to mix its own windows with windows opened by local clients on the real X server. The -R command line option sets the rootless mode for nxagent.

Concerning the original Xnest, distributed by the XFree86 project, NoMachine has implemented significant improvements. By integrating frame-buffer functionalities, we have eliminated many GetImage requests that were needed by the original Xnest. A lot of work was devoted to font handling in order to solve the problems you can find described in the Xnest man page.

nxagent implements clipboard integration with the host X server and internal handling of expose events. This last feature permits nxagent to greatly speed up repaints as it does not have to wait for expose events coming from the real X server.

nxagent can be considered a better Xnest and it is, infact, able to run much faster than Xnest, even without X compression functionalities provided by other NX components.

The Xcompext library is used to tie nxagent together with the compression capabilities of Xcomp/nxproxy. Xcompext adds some Xlib functions and some primitives to the X protocol that let the agent compress images and perform X protocol operations in a way that helps Xcomp carry out its job.