SECURE REMOTE OPERATIONS OF NSLS BEAMLINES WITH (FREE)NX

Zhijian Yin, Peter Siddons, BNL, Upton, NY 11973, USA

Abstract

In light source beamlines, there are times when remote operations from users or collaborators are much desired. This becomes challenging, considering cybersecurity has been dramatically tightened throughout many facilities. Remote X-windows display to Unix/Linux workstations at the facilities, either with straight X-traffic or tunneling through ssh ("ssh -XC"), is quite slow over long distance, not quite suitable for remote control/operations. We implemented a solution that employs the open source FreeNX server. With its efficient compression and proxy server to reduce X round-trip traffic, the bandwidth usage is quite small, and the response from long distance is very impressive. The setup we have, involves a FreeNX server configured on the Linux workstation at the facility's end station. Remote users can use free downloadable clients (Windows, Mac, Linux) at the remote site to connect to the FreeNX servers, through the ssh gateway at the lab. All traffic are tunneled through ssh, and the response time is good enough that remote operations are routinely performed. We believe this technology can have great implications for other facilities.

INTRODUCTION

Remote operations have been a goal for light source users, ever since the internet revolution in the 90s. Many methods has been tried, for example, web servers, together with Java programing has been tried for remote users to monitor and control the experiments at light source beamlines. With the tightening of cybersecurity and the general requirement that web servers not to open to the public unless absolutely necessary, this mode of operation is getting more and more difficult.

X-Windows was developed with networking in mind. As a matter of fact, X-Windows is a network protocol. Within short distance, the X-Windows remote display works well, which is also true with the secure version of remote access, SSH. With compression, "ssh -XC", X display works satisfactorily within short distance. Researchers routinely use X-windows remote display to run applications remotely.

It's a different story for longer distance, which is a situation we have to deal with when we intend to do remote operations for facility users or collaborators who are in other institutions. Network latency caused by media delay, switch/router delays etc., makes X remote display painfully unresponsive. The reason behind this is that X protocol involves many round-trip "checks and balances" when displayed remotely. The large bandwidth usage together with the accumulation of network latencies makes response very slow.

A group of software developers at NoMachine.com invented an ingenious solution to the problem. The main ideas are to compress X protocol traffic using differential compression and more importantly, to reduce the number of X protocol round trips across the network by using a proxy X server. The differential compression algorithm is smart enough to discard useless information, cache messages that could be used later and use different compression scheme for images and text. For detailed technical information, see NX documentation.[1]

Once NX server and clients are made to work, many people are amazed how responsive it is and how little bandwidth it uses. With that kind of responsiveness, it's not hard to imagine using it for remote monitoring, control or operations. Below we will describe our implementation at many of the NSLS facility beamlines.

CONTROLS AT THE NSLS FACILITY BEAMLINES

In many of our facility beamlines, we employ Linux workstations with X-Windows applications for the control of beamline optics and experimental end stations, as well as data acquisitions. The control system is based on the popular EPICS[2] toolkit. A VME based single board CPU (typically Motorola MVME5500 or MVME230x) running EPICS IOC server applications on top of the open source real-time operating system RTEMS[3], is used to drive the hardwares (VME controllers for motors, scalers, A/D, etc.). EPICS clients (medm screen[4], SPEC[5]) at the Linux workstations, communicating with the EPICS IOC server through Channel Access, provide the interfaces for users or beamline operators to control their instruments or experiments.

For security, the control network is isolated in a private LAN, with the Linux workstation containing dual network interfaces, eth0 to the public network, and eth1 to the controls network. Per agreement with the IT security division at the lab, there is no routing between eth0 and eth1. To access the beamline's control network, or to gain access to the control to the beamline, one needs to login to the Linux workstation, which is administered by the local beamline scientist or IT personnel.

NX SERVER, FREENX AND NXCLIENTS

To set up the NX server on the Linux workstation, one have two choices: (1) Download the binaries from www.nomachine.com, where the personal server (two simultaneous connections) were made free some time ago; or one can purchase the commercial version, with technical support. (2) Use FreeNX, a package containing a suite of shell scripts using the core NX libraries, which nomachine.com GPLed and donated to the public domain. At the time when we were experimenting the NX server in 2005, the free binary server from nomachine.com was

not available (only a demo version was), but we got the FreeNX working.

FreeNX can be downloaded from many sources. It comes with binaries in many flavors of Linux, as well as source code. In our beamlines, where Debian distribution is used, FreeNX installation and configuration is straight forward: with Advance Package Tool (apt) in Debian, other software packages which FreeNX depends on are installed automatically, which mostly include "ssh", "expect" etc.

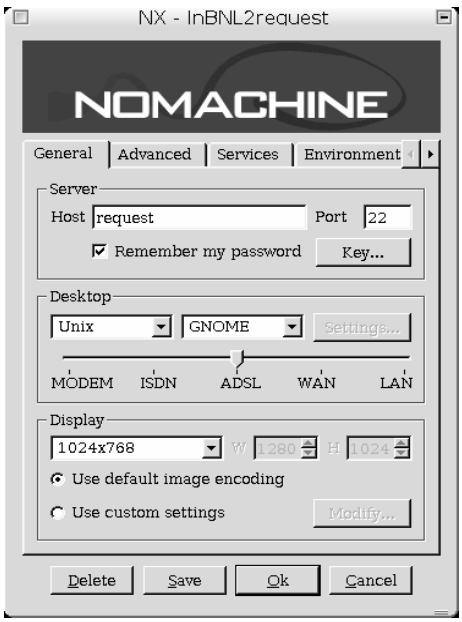

Figure 1: nxclient configuration.

On the remote user's end, they just need to download the "nxclients" software from nomachine.com. Nxclient is close-source software, but it's a free (as in beer) download. It's available in the popular platforms: Windows, Mac, and Linux. Configuring the nxclient is also easy, fill in the hostname and port number for ssh (normally this is "22"), and one is ready to go. Connections types (MODEM-ISDN-ADSL-WAN-LAN) can be chosen to determine the compression level. In the office environment, or at home with cable modem broadband, the default value of ADSL works fine. Once the configuration is saved, one is at the login window, where "username" and "password" are used for authentication.

CYBERSECURITY AT BNL AND SSH PORT FORWARDING

The above arrangement (FreeNX server and nxclient) works quite well inside of our laboratory. Scientists and engineers liked the responsiveness, and the familiar "gnome" or "KDE" windows manager (chosen at the nxclient configure screen), and they start to use the setup, prefer it over the standard "ssh -XC". To do real remote operations from outside the laboratory, for example, from staff's home or users in their home institutions which could be hundreds of miles away, where NX technology is to shine, one more “obstacle” has to be overcome: cybersecurity.

At BNL, as in many institutions, cybersecurity has been dramatically tightened in the last couple of years. The current cybersecurity perimeter defense configuration at BNL involves firewalls, gateways, proxy servers. The only way for staff or users to access is VPN (staff only) or ssh gateway accounts. Thus, a user intending to remotely access the beamline computers has to login to the ssh gateway machine first, and from there, ssh to the beamline computer. It's a two step process.

Thus we face a problem: the nxclient needs a "hostname" and "port number". With the perimeter defense, most of the hostnames at BNL are not routable or resolvable. Fortunately, ssh has a "port forwarding" feature,[6] also known as "ssh tunneling", to map the remote site tcp port (e.g., 22 for ssh in our case) to a localhost port via the ssh gateway. An example in Linux: "ssh -L localport:remotehost:22 username@ssh_gateway". In Windows or Mac, one just need to configure ssh port-forwarding for the ssh client. With this tunnel set up, one can configure the nxclient to use "localhost" and "localport".

PUTTING IT TOGETHER: REMOTE OPERATIONS WITH FREENX

Once logged in, the user is presented to the familiar "gnome" or "KDE" window manager, just like login locally. All the features are available, and almost all X-Window applications run fine. The screen snapshot (Fig. 2) shows a remote window, with EPICS medm displays for beamline motors and scalers. The great advantage of NX, compared with other remote X-servers (e.g., exceed), is the responsiveness over long distance. In a typical remote login from home, with cable modem broadband, the authors have no problem to run experiments at the beamline.

The FreeNX/nxclient combination has been made to work successfully at NSLS since December 2005. It provides a powerful tool for beamline scientists and engineers to remotely diagnose and troubleshoot instrumentations, and users and collaborators for remote operations. For example, just recently a scientist (one of the authors, P.S) was in Australia on a business trip when one of his prototype detectors in use at one of the NSLS beamlines got reset due to power loss. Users were confused and emailed the scientist for help. The scientist was able to remote login with nxclient, displayed most of the detector parameters with graphical tools (medm screen, similar to above) and in a snap, set the correct parameters from thousands of miles away.

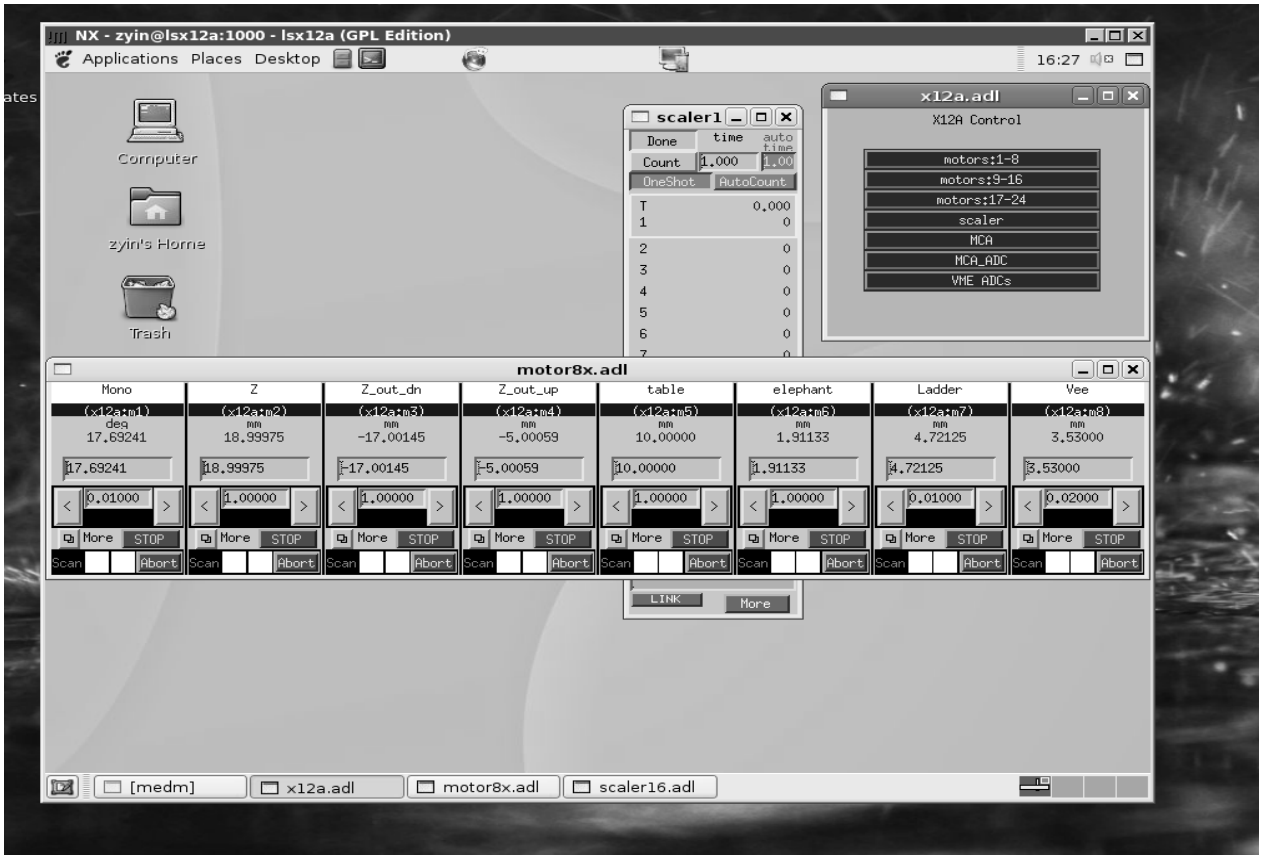

Figure 2: Snapshot of remote screen.

CONCLUDING REMARKS

Smart and efficient compression, together with proxy X server and other tricks make NX the most responsive remote desktop for X-Windows system today. Taking advantage of the great responsiveness of the NX technology, as well as port-forwarding feature of SSH, we successfully implemented FreeNX for remote beamline diagnostics, data processing and operations at NSLS. As all traffic are tunneled through SSH, these operations are secure. If so desired, additional security keys can be generated and dispatched to authorized users. The solution is generic, no programming is involved. We believe this solution could have great implications for many facilities.

ACKNOWLEDGMENTS

We thank the developers at nomachine.com, and the FreeNX package developer and maintainers for their great work. We'd also like to thank staff and colleagues at BNL for testing and feedback. This work is performed at the National Synchrotron Light Source, Brookhaven National Laboratory, which is supported by the US Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-98CH10886.

REFERENCES

[1] http://www.nomachine.com/
[2] http://www.aps.anl.gov/epics
[3] http://www.rtems.com/
[4] EPICS medm extension: http://www.aps.anl.gov/epics/extensions/medm/index.php
[5] www.certif.com
[6] SSH user's manual