Virtual Network Computing
REMOTE CONTROL

Virtual Network Computing (VNC) allows different computers to access a common desktop. Hans-Georg Esser explains the ins and outs of VNC and how to make it more secure with an SSH tunnel

Those who regularly work on more than one computer know the procedure: after logging on, you have to open windows from scratch, load documents again and find your old Web pages. Modern desktops have session management features, so that applications on associated desktops at least reappear in the correct place. This is however not possible with "non-desktop" programs such as Netscape.

It gets even worse when two assigned computers run on different operating systems. A good network set up can make sure that you can access your Linux home directory from Windows, but all the normal applications will be missing.

An elegant solution

VNC solves the problem by taking the complete desktop from one computer and reconstituting it in a window on a second computer. In this article, we will describe the setup of a VNC server under Linux as well as the setup of a VNC client under Linux and Windows.

Server

A VNC server under Linux is in reality a double server. On the one hand it can be considered an X server – you can usually access it through *localhost: 1* after startup. This however will do nothing, as the server will be running without an output. In order to see the desktop that is running on the new X server, you have to start a VNC client (*vncviewer*). The VCN client then plays the role of the VNC server and transfers the desktop contents as picture information.

To set up the VNC server, you either need the VNC source texts or a precompiled rpm package named *vnc server*. For the latter case, the installation is a simple:

```
rpm -Uvh rpm-server.....rpm
```

Some distributions provide the *vnc* and *vnc server* packages already, and in this case it is simplest to install these. For other distributions, you can find these packages on this month's coverdisc. The sources are also available from AT&T's download page (*http://www.uk.research.att.com/vnc /download.html*).

Server start

The next step is the starting of the server. If you want to first test the settings of the VNC server, simply call up the *vncserver* script – this then starts the actual server, *Xvnc*, with the standard parameters. Pay attention to the output of the server start script. A display number is displayed here (: 1, : 2, etc), which you will need in order to access the server.

For security reasons, you now need to specify yet another password for access. To do this, enter the command *vncpasswd* and type your desired password (which has nothing to do with your account password) twice. It is then coded and entered in the file *~/.vnc/passwd*.

Linux client

For a first test, locally access the current VNC session. Install the *vnc* package that contains the *vncviewer* client. If you now enter the command:

```
vncviewer local host: 1
```

in the console (where ": 1" is replaced by the correct display number where necessary), the new desktop will appear in its own window. Tip: make sure that the VNC password is correctly entered. The window manager is an old Unix WM. *twm* is not the easiest to use, and you will therefore probably want to change it.

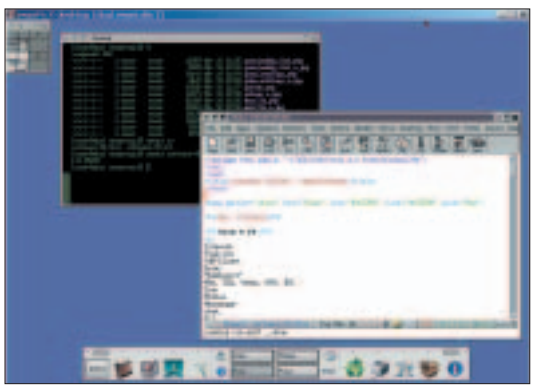

Figure 1: A VNC viewer under Windows (identifiable by the frame of the window) indicates a Linux desktop with an xfce window manager

Adapting the desktop

The VNC server implements the *~/.vnc/xstartup* script after start up. This contains the line *twm &*, which starts the *twm*. It is possible to simply replace this line with one that starts a desktop of your choice, for example *startkde &*, *startgnome &* or *startxfce &* (for the XFCE desktop, which comes with a small memory).

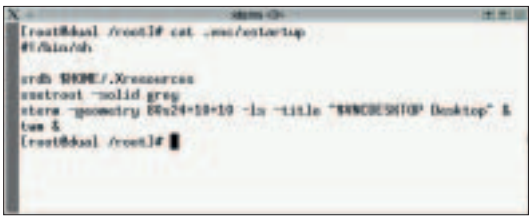

Figure 2: The window manager is also specified in the ~/.vnc/xstartup file

Several clients

Trying to start a second *vncviewer* process will be only partly successful. After entering your password, a new window with the new desktop will open, but the first window will however be closed. The reason for this is that you started the clients in the "non-sharing" mode. This means that VNC displays can only be shown by one client at a time.

If you want to work at two workstations (for instance in two offices or areas in the same building) at the same time, then a divided access is for you. The simplest way to achieve this is to start the server in the "always shared" mode of operation. The following command is used for this purpose:

```
vncserver -kill: 1
```

(The display number can again be adapted if necessary.) The server should then be started again, this time however with the following command line:

```
vncserver -alwaysshared -geometry 1000x700
-depth 24
```

This example has another two common arguments in addition to the "–alwaysshared" option for the sharing mode:

- *–geometry 1000x700* defines the size of the VNC desktops, in this case 1000 x 700 pixels.
- *–depth 24* defines the depth of colour. The server runs with a mere 8 bits as standard, which doesn't look too good on higher resolution monitors.

Start *vncserver* again. If there has not been enough time since the last interruption, the server will not be able run on the same port and will take another. This will then also change the display number (: 2 instead of : 1), which must be taken into account when performing new client starts. You can therefore have as many simultaneous connections as you want with the altered set up.

The whole thing makes sense when you access the server from another computer. If the computer is a Linux PC, it is a good idea to install the client there (the server is not necessary). In place of the *vncviewer local host: 1* command, you need only enter the name of the computer. For example, you would enter:

```
vncviewer myserver: 1
```

if the computer on which the VNC server runs is called *myserver* and desktop number : 1 is used. As a result, you will now see the same desktop on two computers.

Network rush hour

If you only set up one local connection, the data between the client and the server will be transferred in an uncompromised (raw) form. This would be a problem in a network, which is why VNC uses compression algorithms for network connections. In a test connection over DSL, the time required for screen actualisation was acceptable; this does not apply to slower modems or ISDN connections. Nevertheless, if you want to try it with a modem, a small trick will be helpful: Simply use a very small desktop (for example 500x400 pixels) and only 8 bit depth of colour – this should be enough to display a smaller application.

Windows client

In a heterogeneous environment, you will perhaps want to access the VNC desktop from a Windows computer. To this end, you will use the VNC Windows client, which we have also included on the coverdisc. After installation (typical to Windows), you will find an icon on the desktop with which you can call up the viewer. You will be asked for the server name in the form of "computer name: display", thus "myserver: 1". If you cannot find out the name of the local network on your Windows computer, use instead the IP address such as "192.168.1.199: 1".

Long live the desktop

If the VNC server runs on a computer that is not switched off, then you can also keep a VNC session eternally open. Even when all VNC clients are logged off, the VNC server still remains active, and all programs started under it will continue to run. Thus when you log on again, you will come to the exact same state as when you ended the client. For security reasons, you should however save all open documents before you close the client.

Security with SSH

VNC has a password that protects against the establishment of unauthorised connections – the actual transmission is however unencrypted (similarly

to a Telnet session). If this isn't secure enough for you, you can create an SSH tunnel.

To do this, firstly start the VNC server with the additional option *–localhost*. In this way, no connections can be established by other computers. As your intention is to create an access from another computer, you have to "tunnel" into the VNC port through SSH. That can be done by firstly calculating the port number of the VNC server by simply adding 5900 to the display number – thus display: 1 corresponds to a port number of 5901. Next, enter the following SSH command:

```
ssh -L 5901: myserver: 5901 myserver
```

This assumes that the VNC server runs on a computer, *myserver*, and that you are entering the command at a client computer. SSH will now ask for the password as usual. Another thing that happens is that connections to port 5901 at the client are forwarded to port 5901 at *myserver*. If you now want to start the VNC Client, enter:

```
vncviewer localhost: 1
```

The viewer then looks for a VNC server that is running locally and finds the local port 5901. This is encoded and passed on by SSH to the correct port on the server, thus maintaining the VNC connection. If the client is a Windows computer, you will have to install an SSH package for Windows – a description of which is beyond the scope of this article. The SSH command is then identical.

krfb: VNC for KDE with more features

Some of you, while reading this, may have had the thought: "Why do I have to create a new desktop and not just use the active one?" This is where the *krfb* project comes into play. If you translate and start the program from the sources, a new icon will appear in the KDE panel (KDE 2.2 and 3.0 are supported), which allows the configuration of the VNC server. Select a desktop number (e.g. 1, to be able to address the desktop as "server: 1") and enter a password.
If you now start the VNC Client on another computer (not on the same one!), a direct connection will not automatically be established. What will happen is that a *krfb* window will open on the server, which will inform you that a client would like to establish a connection. If you permit the establishment of the connection, you can then enter the password at the client computer. And hey presto, you can see your active KDE server desktop in the window. This therefore does away with the need to start a new (empty) desktop, but you can still work on the distant computer with the normal desktop. This feature is not offered by the regular VNC server. In the VNC window, you can even switch between the different KDE "desktops" by using Ctrl+F1, Ctrl+F2 etc.

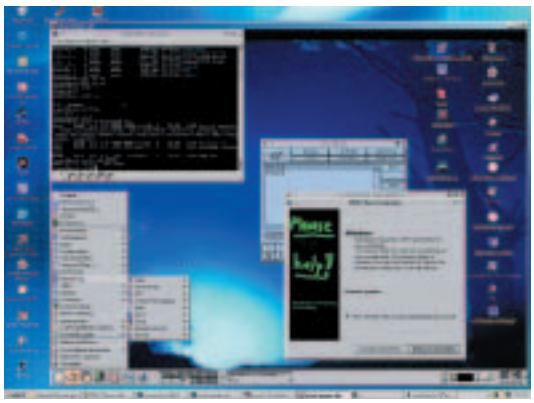

Figure 6: Illustration of a KDE desktop under Windows, exported by krfb (picture = 1600x1200 pixels, size: 390 KB)

Compiling krfb

Here is a short installation guideline, as there are no rpm packages for *krfb*. Use the *kfrb* version 0.5.1 for KDE 2.2.x, and the version 0.6 for KDE 3.0. Unpack the source code archive from this issue's CD, change to the newly created *krfb-0.x.x* directory and, here, carry out the usual Linux command lines as root user.

```
/configure
make
make install
```

The last step installs all created files below */usr/local/kde/*. Then simply start the client by entering:

```
/usr/local/kde/bin/kfrb &
```

If you would rather install *krfb* into the KDE standard directory (*/usr* or */opt/kde2* depending on the distribution), use the option – *–prefix=/usr* (or – *–prefix=/opt/kde2*) when doing the *configure* step.

Those who don't work with KDE, but want to be able to access the normal X server, should check out the X0rfbserver project. The software fulfils the same function as *krfb*, but doesn't need any special desktop system such as KDE or GNOME.

The VNC Web site, *http://www.uk.research.att.com/vnc/*, offers a lot of additional information on VNC, including a comprehensive FAQ document amongst other things.

Info

VNC homepage: *http://www.uk.research.att.com/vnc/*
Direct VNC, a VNC client which doesn't run under X, but on a Linux Frame buffer (console): *http://www.adam-lilienthal.de/directvnc/*
krfb, KDE VNC server: *http://www.tjansen.de/krfb/*
X0rfbserver: *http: hexonet.de/software/x0rfbserver/*
CygWin Tools for Windows, including Bash, ssh etc: *http://sources.redhat.com/cygwin/*