Remote Working and Virtual Desktops at the AIfA

The following assumes you have a home machine ("laptop") and a desktop machine at the AIfA ("desktop"). You can always – of course – connect direct from your desktop to itself. Last update 10th January 2014.

Quick route to success for the lazy:
1) Set up password-free access (Sec. 1.1).
2) Set up your `.ssh/config` file to get through the gate machines (Sec. 1.3).
3) If you use *SVN* (or *git* or equivalent) on your PC, which you should for everything except big datasets, automatically access it by following Sec. 1.3.1.
3) If you use *NX*, which you should for remote desktops, set it up to automatically tunnel (Sec. 1.6).

1 Remote login

You always have to connect from your laptop to your desktop. SSH is the standard tool for this, you should not (can not?) use anything else. See e.g. www.openssh.org

1.1 Password free

If you have to enter a password to access your laptop, why use it again to get to the AIfA? Just set up a secure key and never use a password to access the AIfA again.

- Pros: Great for scripting, makes life easier.
- Cons: If your laptop is stolen and you haven't locked the screen or logged out, anyone can access your AIfA machine and data.

Detailed instructions to be found at:
http://www.thegeekstuff.com/2008/11/3-steps-to-perform-ssh-login-without-password-using-ssh-keygen-ssh-copy-id/
Typically I would run:

```
ssh-keygen

ssh-copy-id -p 1234 izzard@gate1.astro.uni-bonn.de

ssh gate4.astro.uni-bonn.de

ssh -p 1234 gate1.astro.uni-bonn.de
```

1.2 Tunnels

If you want to connect from your laptop to your desktop, you have to go through one of the `gate` machines (`gate`n`.astro.uni-bonn.de` where $n = 1,2,3,4$) using `ssh`. Note that gates 1-3 use port 1234 for `ssh`, gate 4 uses port 22 (which is the `ssh` standard port).

In this section I explain how to tunnel to your PC, but you *do not need to do this manually*. For a (perhaps) better solution see Section 1.3 below.

e.g. To connect port 23456 on your laptop to port 6666 on `aibn36`, use the following command in a terminal (and leave the terminal open!)

```
ssh -p 22 -g -t -L 23456:localhost:23456 izzard@gate4.astro.uni-bonn.de
"ssh -g -L 23456:localhost:6666 izzard@aibn36.astro.uni-bonn.de"
```

note that there is *no line break* in the above command.

Following this, port 23456 on your laptop is *directly connected* to port 6666 on your desktop. Anything running on your desktop will "hear" on port 6666 whatever your laptop sends on port 23456. You have to match port numbers carefully.

Explanation

The first part of the command

ssh -g -t -L 23456:localhost:23456 izzard@gate4.astro.uni-bonn.de

connects your laptop ("localhost") to `gate4`, forwarding port 23456 on your laptop to port 23456 on `gate4`. Then, the part in quotes

"ssh -g -t -L 23456:localhost:6666 izzard@aibn36.astro.uni-bonn.de"

is the command executed on `gate4` to forward `gate4`'s port 23456 to port 6666 on aibn36. Note that when gate4 executes this command, the "localhost" is actually *gate4* and not your laptop. Why? Because the command is being run *on gate4*...

If you are using putty you will have to work out how to do this yourself. It is not difficult, there are many guides online, e.g.
http://robmd.net/blog/tunneling-with-putty.html

1.2.1 Keeping tunnels open with `autossh`

Tunnels tend to end when their `ssh` connection times out. This is annoying, or a good security measure, depending on your point of view. Consider using `autossh` in the above commands instead of `ssh`. `autossh` is a program to start a copy of `ssh` and monitor it, restarting it as necessary should it die or stop passing traffic.

1.3 Automated Tunnels

You can automate the setting up of SSH tunnels by putting some lines in your `~/.ssh/config` file. The following configuration allows `ssh` to *automatically* access the gate machines, and my PC (`aibn36`), with a simple "`ssh`" command. The sections for each gate machine define how to access it (i.e. the access port). Note that if I try to `ssh` to "`gate1.astro.uni-bonn.de`" then I have to specify the port myself: only access to `gate1` (without the `.astro.uni-bonn.de`) is automatically via port 1234. Also note that *only* `ssh` knows about these "aliases".

The final section defines a new host, aibn36, and sets up the "ssh proxy" to tunnel through `gate4`. This is identical to the tunnelling described above, but is automatic. Note that I have turned on X11 forwarding so I can run remote terminals, or otherwise.

```
Host gate1
        Port 1234
        Hostname gate1.astro.uni-bonn.de
Host gate2
        Port 1234
        Hostname gate2.astro.uni-bonn.de
Host gate3
        Port 1234
        Hostname gate3.astro.uni-bonn.de
Host gate4
        Port 22
        Hostname gate4.astro.uni-bonn.de
Host aibn36
        Hostname aibn36.astro.uni-bonn.de
        ProxyCommand ssh -q -W %h:%p gate4
        ForwardX11 yes
```

1.3.1 Example: Automated tunnels with SVN

I have an *SVN* server running on `aibn36`, so in order to use it I have to tunnel through the gates. This would normally mean opening a tunnel and *leaving it open*. This is a pain (see Section 1.2.1 above). You don't have to. With the above setup, just check out your *SVN* repository using the `svn+ssh` method remembering to use the host name given in your `.ssh/config` file e.g. when I check out my "tex" repository, I use:

```
svn co svn+ssh://aibn36/users/izzard/data/svn_repos/tex
```

Note: if you used to have your *SVN* tunnelled manually, you have to migrate from `svn://` to `svn+ssh://`, e.g. with

```
svn relocate svn://localhost:10705 svn+ssh://aibn36
```

where 10705 is the port that used to be used for connection. You can determine which you're using with `svn info`.

1.4 Extra SSH configuration

I have some extra configuration in my `~/.ssh/config` file (where ~ is your home directory, both on the laptop and desktop, assuming you're using Linux/Unix) which may be useful for preventing timeouts.

```
Host *
ControlMaster auto
ControlPath ~/.ssh/cm_socket/master-%l-to-%r@%h%p
ServerAliveInterval 5
ServerAliveCountMax 99999
TCPKeepAlive yes
```

1.5 `.ssh/config` permissions

You should make sure only you can read and write your `.ssh/config` file. To do this, run

```
chmod 600 ~/.ssh/config
```

1.6 NX *with* ssh *tunnels*

If you try to use *NX* to communicate with your desktop, it will *not* use the tunnels in `.ssh/config` by default. This is because *NX* uses its own version of `ssh` called `nxssh` which ignores your `.ssh/config`! Annoying as this is, there is a workaround which you can use on your laptop or anywhere you have root access (see http://www.nerdenmeister.org/2013/11/18/nx-through-proxy-tunnel/). Assuming you're on Linux, do the following:

```
cd /usr/NX/bin
sudo mv nxssh nxssh.bin
```

Then open a file, as root, called `nxssh.csh` e.g. with your favourite editor (which is emacs of course),

```
sudo emacs nxssh.csh
```

into which write the following

```
#!/bin/csh -f
set params = ( )
@ i = 0
while ( $i < $#argv )
@ i++
if ( "$argv[$i]:q" == "-E" ) continue
set params = ( $params:q $argv[$i]:q )
end
exec $0:h/nxssh.bin $params:q
```

Save this and close the editor.

Make this new file executable and symlink it to `nxssh` (so that the script is executed instead of nxssh)

```
chmod a+x nxssh.csh
sudo ln -s nxssh.csh nxssh
```

Now, when NX is run you can access your PC (e.g. `aibn36`) *directly* and on port 22 (the `ssh` port) just as you would if there were no `gate` machine! Hooray!

2 Files with sshfs

With this solution, your desktop's disk is mounted on your home machine. In effect, your desktop behaves similarly to a USB disk plugged in to your laptop, except that instead of transferring data over a USB cable, it is transferred over the network.

- Pros: Direct access to all your data. Applications run on your laptop and use your laptop's CPU power. No tunnel is required because the gates can access your desktop disk directly (through NFS).
- Cons: Data transfer might be slow.

To mount aibn36's disk in your laptop's `/tmp/aibn36` directory, use:

```
mkdir /tmp/aibn36

sshfs izzard@gate4.astro.uni-bonn.de /vol/aibn36/aibn36_1/izzard /tmp/aibn36
```

To unmount (do this before shutting down your laptop):

```
fusermount -u /tmp/aibn36
```

3 Virtual desktop

In this solution, you set up a virtual desktop on your desktop PC. This looks just like your true desktop, but is both *persistent* (i.e. you can log out of your desktop and it is still available) and *accessible from both your laptop and desktop.*

- Pros: Uses storage and CPU of your desktop. You laptop is just an interface. Bandwidth can be controlled (by reducing colour depth, for example).
- Cons: Requires an ssh tunnel through a gate[1]. Speed depends on your home network bandwidth and choice of settings (see Sec. 5.1).

All virtual desktops rely on a client–server connection, which must be made through the gate. In some cases you have to run the server yourself, in some cases it is automatically run for you.

[1] Although with Remmina this is handled automatically!

3.1 VNC

VNC (virtual network computing) sets up an X window session for itself through the `vncserver` command, which you can then access with the `vncviewer` command. The best VNC for use with a modern desktop is *Tiger VNC* sourceforge.net/projects/tigervnc/ but also consider *Tight VNC*, from www.tightvnc.com. (Tiger is not installed but you can download the executables and run them yourself, root is not required.)

Pros

- Old tech, reliable, available on every platform (also Android).
- Bandwidth control, e.g. colour depth, compression, can be changed on the fly.
- Desktop can be shared with other users.
- Open source.
- Can grab your true desktop with x11vnc

Cons

- No 3D. Command line tools (but perhaps not on Windows/Mac?).
- Tiger VNC is not installed by default at the AIfA.

Instructions

- On your desktop (the *server*) run

```
vncserver
```

There are numerous options which can be specified to the server, e.g. the screen size `-geometry` *widthxheight* and colour depth `-depth` *depth*.

When `vncserver` runs, it tells you the number of the X display it sets up (e.g. *1*, *2*, ...). You need to remember this number. If you have a normal desktop running, this is usually *1* (your normal desktop is display *0*).

- On your laptop (the *client*) launch in one terminal the ssh tunnel (and leave this open!) where the destination port *5901* should be *5900* plus the number of the display given by vncserver (in this case display *1*)

```
ssh -S none -p 22 -g -t -L 20792:localhost:20792 izzard@gate4.astro.uni-bonn.de "ssh -S
none -g -L 20792:localhost:5901 izzard@aibn36.astro.uni-bonn.de"
```

(As before, this is a single line command!)

- Set up password with `vncpasswd` command (beware, stored in plain text, but ssh connection *is* secure)
- In another terminal run the vncviewer

```
vncviewer -ZlibLevel 9 -LowColourLevel 2 -PreferredEncoding "ZRLE" localhost::20222
```

where the port number (here 20792) must match that chosen for your ssh tunnel.

3.2 NX

NX is a VNC alternative with many client/server packages, the most popular is www.nomachine.com which is installed at the AIfA. See also https://www.linux.com/learn/tutorials/392935-remote-linux-desktops-with-nomachine-nx

Pros

- No manual run of the server is required.
- Bandwidth selection (from MODEM to LAN speeds).
- Uses port 22 always.
- Web plugin to interface with your desktop through a browser.
- Graphical user interface.
- 3D support.
- Sound! Assuming you use pulseaudio, enable the networking options with the *paprefs* command: "Make discoverable PulseAudio network sound devices available locally", "Enable network access to local sound devices" and "Don't require authentication".
- Printing (CUPS) and Samba shared directory support (UNTESTED)

Cons

- Old-fashioned user interface.
- Not free: 3 user limit.
- Display size is fixed once set (but you can change it with `xrandr -s` *`x`*`xy` where `x` and `y` are the new resolution)
- No desktop sharing.
- Sound *requires* the use of the antiquated *ESD*.
- Closed source.

Instructions

- Server

You don't have to do anything once it is installed.

- Client: ideally use the automated ssh tunnels described in Sec. 1.3 and Sec. 1.6.

Otherwise do the following to make your own ssh tunnel, e.g. here on port 20793:

```
ssh -S none -p 22 -g -t -L 20793:localhost:20793 izzard@gate4.astro.uni-bonn.de "ssh -S none -g -L 20793:localhost:22 izzard@aibn36.astro.uni-bonn.de"
```

run the client software

```
/usr/NX/bin/nxclient
```

You then have to configure the client. You should set the host to be *localhost* and the port to be *20793* (or whatever you have chosen when making the ssh tunnel). You can choose your Desktop type (e.g. KDE, Ubuntu, Gnome) and display size. The bandwidth setting should match your connection.

3.3 X2GO

x2go wiki.x2go.org is an NX fork and hence has most of the features of NX. In addition, the GUI is superior, automatic desktop resizing is supported and sound works more reliably.

Pros

- No manual run of the server is required.
- Bandwidth selection (from MODEM to LAN speeds).
- Uses port 22 always.
- Web plugin to interface with your desktop through a browser (install *x2goplugin*).
- Graphical user interface (superior to NX).
- 3D (you must comment out `the line X2GO_NXAGENT_DEFAULT_OPTIONS+=" -extension GLX"` in `/etc/x2go/x2goagent.options`)
- Sound! Assuming you use pulseaudio, enable the networking options with the *paprefs* command: "Make discoverable PulseAudio network sound devices available locally", "Enable network access to local sound devices" and "Don't require authentication".
- Sound supports *pulseaudio*.
- Printing (CUPS) and Samba shared directory support (UNTESTED)
- Automatic resizing of desktop.

Cons

- Occasionally the connection fails to start. Just retry a few times and it should work and/or try restarting the ssh tunnel.
- Not yet installed on all the AIfA machines (just ask `ticket@astro.uni-bonn.de`)
- No desktop sharing.
- Minor bugs.
- Major bug! Sometimes closing the connection by closing the window *Terminates the session!* This is a fatal flaw for me.
- Open source.

Instructions

- Server

As with NX, you need to do nothing.

- Client

First open the ssh tunnel, in this case to port 20794

```
ssh -S none -p 22 -g -t -L 20794:localhost:20794 izzard@gate4.astro.uni-bonn.de "ssh -S
none -g -L 20794:localhost:22 izzard@aibn36.astro.uni-bonn.de"
```

Then run the client

```
/usr/bin/x2goclient
```

Setup is very similar to NX: You should set the host to be *127.0.0.1* (***NOT localhost***, if you do it will work only some of the time!) and the port to be *20793* (or whatever you have chosen when making the ssh tunnel). You can choose your Desktop type (e.g. KDE, Ubuntu, Gnome) and display size. The bandwidth setting should match your connection.

3.4 Remmina

http://remmina.sourceforge.net/

Remmina is a GTK+ viewer which can use NX, RDP (remote desktop protocol), VNC and a few other types of connections (but not x2go). If you want to connect through an ssh tunnel in the easiest way possible, this may be the tool for you!

Pros

- Simple GTK+
- Many protocols (NX, RDP, VNC...)
- Only runs on Linux
- 3D
- No dodgy terminate button which you can accidentally press!
- SSH tunnelling is automatic once set up (which is very simple)

Cons

- Can something not running on Windows be considered bad? If you run Windows you have to live with yourself...
- The window sometimes does not refresh correctly. Resize the window (or jump to full screen mode and back) and it will.
- CTRL-ALT-F for changing from full screen to a normal window crashes Remmina. Solution: Move the mouse to the drop-down menu in the top middle of the screen and use the icon. This does not crash. To prevent this bug, go to "Preferences" in the main window, click "Keyboard" then click the button next to "Toggle fullscreen mode". In the popup window click "Remove" then you cannot make this mistake again!
- Screen refresh seems not to be quite as fast as NX, particularly in terminals (glxgears is ok).

Instructions

Run `/usr/bin/remmina` e.g. from a terminal. Go to the menu *Connection->New* or click on the "New" icon.

Fill in the boxes as follows:

- In "name" enter a name for the connection (e.g. "`work`" or "`aibnxyz`"), you also have to choose a protocol or just use the default NX. Server should be the hostname of your desktop machine (NOT localhost or 127.0.0.1). User name should be obvious. You can give it a password, or it will prompt later. Quality as usual depends on your connection (try them!). "Startup program" allows you to choose KDE or GNOME or XFCE.
- Next click in the SSH tab. Click "Enable SSH tunnel" and click "Custom". Enter (e.g.) `gate4.astro.uni-bonn.` and fill in your user name. If you have set up the public key (passwordless) login as explained above, select "Public key (automatic)" otherwise "Password".

- Now press “Save”.
- The server “`work`” should now be in the main list of connections: double click it. You will now be *automatically* tunnelled through to your new remote desktop server. You can start a new desktop, resume an existing one, etc., as with NX.

Note: in Preferences->Resolutions you may want to add your screen resolution. I have no idea what this list does!

3.5 Conclusion

I am currently using Remmina with NX by default. Why? Pure SSH is very inefficient. sshfs is fine for files only but is not persistent (I love my virtual desktop that I can access from *anywhere*). VNC is slow, old and lacks 3D, and while it is quite stable it is time to move on. NX (with nomachine) is fast and stable, but the GUI is annoying (just accidentally terminate and see how much you swear!). x2go is very pretty and very fast, but unreliable with connection problems. Remmina is stable, fast, uses many protocols (including NX) and has automatic SSH tunnelling. One click, enter your password, and you’re done. How much easier can it get?

4 x2go/NX notes

4.1 Sound

Ubuntu uses pulseaudio, so if you want to use NX to play sound you have to install the esound (ESD) compatibility layer and even then I couldn't make it work. Use x2go instead, because it natively supports pulseaudio.

Remember to run *paprefs* and enable the following: "Make discoverable PulseAudio network sound devices available locally", "Enable network access to local sound devices" and "Don't require authentication".

4.2 Videos

VNC is just too slow for video, but NX and x2go work fine.

4.2.1 Sound driver

You will have problems if you want to play sound with NX (see above) but if you want to avoid this just run mplayer (for example) with

```
mplayer -ao null ...
```

of course this means you have no sound! Better to use x2go which just works.

4.2.2 Video driver

If your video is slow, try

```
mplayer -vo x11 ...
```

instead of the default xv or gl. This can be *much* better.

4.2.3 Bugs in x2go

x2go is very promising software, but there are a few minor bugs.

- Often there are problems authenticating if you set the host to *localhost*. The workaround is to use *127.0.0.1* instead! Simple. (This reminds me of bugs that often used to plague such programs)
- There is a bug in the Perl script `/usr/bin/x2goumount-session` (on the server and the client). To fix it, change

```
@line[1] e $only_path_windrive
to
@line[1] eq $only_path_windrive
```

4.2.4 Keyboard Shortcuts in NX/X2GO

Keystrokes available in NX 3.x (presumably in x2go as well)

- Ctrl + Alt + Shift + Esc to get rid of a not responding session
- Ctrl + Alt + T to terminate or suspend a session
- Ctrl + Alt + F to switch to fullscreen/windowed (Note: This feature is not available on Windows)
- Ctrl + Alt + Shift + F to switch to multimonitor fullscreen/windowed (Note: This feature is not available on Windows)
- Ctrl + Alt + M to minimize or maximize fullscreen window
- Ctrl + Alt + arrow keys to viewport navigation
- Ctrl + Alt + keypad arrow keys to viewport navigation (Note: this action is performed also by keeping Ctrl + Alt pressed and dragging the content of the main window by the pointer)
- Ctrl + Alt + R to switch "auto-resize/viewport" mode. The agent starts in auto-resize mode, so users can resize the desktop simply by resizing the main window. In viewport mode, resizing the main window doesn't make the desktop resize itself, but users can navigate the desktop by moving the viewport (Note: the auto-resize feature will be available on Windows starting with NX 4.0.0)
- Ctrl + Alt + E to toggle the lazy encoding ***WARNING: crashes x2go***
- Ctrl + Alt + J to force a drawable's synchronization, in order to attempt a fix for a visualization problem
- Alt + F4 when the window manager is present, it allows you to suspend or terminate a session
- Ctrl + Alt + K enable/disable the catching of Alt+Tab and Print Screen keys

5 Comparisons

	homepage URL
native SSH -X	www.openssh.org
Tiger VNC	sourceforge.net/projects/tigervnc/
NX	https://www.linux.com/learn/tutorials/392935-remote-linux-desktops-with-nomachine-nx
x2go	wiki.x2go.org
Remmina	http://remmina.sourceforge.net/

5.1 Refresh rate and bandwidth

I could have used x11perf to test performance, but glxgears is quicker and easier. I did this on my 100 Mbit down [actually about 40-45Mbit today]/5 Mbit up home connection. This is just a guide, of course, but generally:

- Use of "modem" speeds seems pointless. No modem connection is 100+KB/s and the refresh rate is terrible.
- ADSL (medium/good) always gave the fastest refresh rate, and the lowest bandwidth. This is probably the best option at present.
- The "best quality" is always terrible *and* uses high bandwidth. Perhaps it wants even more bandwidth than I have on a Sunday (e.g. x2go) but this doesn't seem to be the case with NX.

	glxgears frames per second	bandwidth use (B=Bytes)
Laptop[2]	2000	N/A
native SSH -X	4-8	3.5MB/s
Tiger VNC	No 3D support	N/A
NX (modem)	1200	165KB/s
NX (ADSL)	1600	250KB/s
NX (WAN)	1000	900KB/s
NX (LAN)	30	1.3MB/s
x2go (modem)	1600	160KB/s[3]
x2go (ADSL)	2000	180KB/s
x2go (WAN)	90	900KB/s
x2go (LAN)	10	4MB/s[4]
Remmina[5] (poor - fastest)	1500	90KB/s[6]
Remmina (medium)	1700	130KB/s
Remmina (good)	1700	150KB/s
Remmina (best - slowest)	14	650KB/s