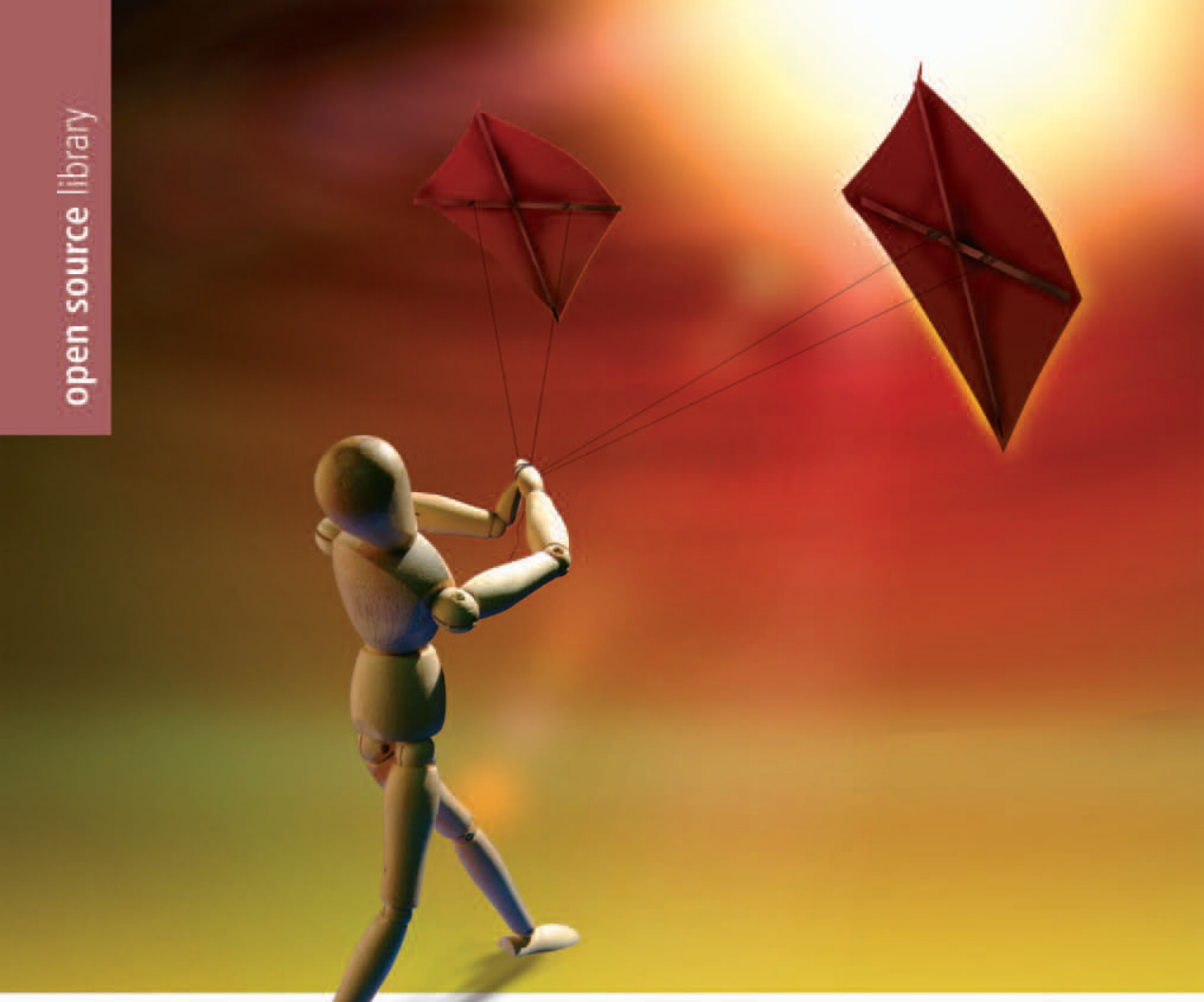
open source library



Michael Gutmann
Detlef Lannert

Linux im Netzwerk

Der Praxisleitfaden für kleine und mittlere Umgebungen

ADDISON-WESLEY

3 Elementare Dienste im Netzwerk

Bisher haben wir uns hauptsächlich damit beschäftigt, was Sie tun müssen, um Ihr Linux-System ans Netzwerk anzuschließen. In diesem Kapitel widmen wir uns jetzt einigen grundlegenden Diensten, die speziell in Unix/Linux-Umgebungen existieren und die Sie von anderen Betriebssystemen in dieser Form vielleicht nicht kennen.

3.1 Mit X im Netz

Netzwerkfähigkeiten sind in X auf einer so grundlegenden Ebene realisiert, dass ohne zusätzliche Software ein Datenaustausch zwischen verschiedenen Rechnern stattfinden kann. Dies stellt jedoch gleichzeitig das Problem dar: Eine Modernisierung des Client-Server-Modells von X ist bisher daran gescheitert, dass die dazu notwendigen Funktionen so tief im X-System verankert sind, dass Veränderungen hier große Auswirkungen auf die Abwärtskompatibilität hätten. Daher findet der Datenaustausch zwischen X-Server und X-Client weiterhin ohne zeitgemäße Verschlüsselung und Datenkomprimierung statt, was das Arbeiten über netzwerktechnisch weite Strecken mitunter stark behindert. Trotzdem stellen die Netzwerkfähigkeiten von X im internen Netzwerk immer noch eine einzigartige Lösung dar, die in anderen Systemen in dieser Form nur durch mitunter enormen zusätzlichen Softwareaufwand zu realisieren ist.

Lassen Sie uns aber kurz die grundlegenden Einstellungen rekapitulieren, die für die Kontaktaufnahme zwischen X-Server und -Client notwendig sind: Zuallererst sollten wir uns vergegenwärtigen, dass der X-Server auf dem Rechner läuft, auf dem sich die Grafikkarte befindet; in einem X-Terminal-vs.-Server-Verhältnis läuft der X-Server also auf dem Client und die X-Client-Software auf dem Server. Aber wenn Sie im Hinterkopf behalten, dass ein X-Server ohne Grafikhardware ungefähr so nützlich ist wie ein Auto ohne Motor, dann werden Sie auch bei diesen komplizierten Beziehungsverhältnissen den Überblick behalten[1].

[1] Wie immer ist dies auch eine eigentlich unzulässige Vereinfachung, da es auch X-Server gibt, die ohne Grafikkarte laufen. Aber erstens ist dies kein X-Buch und zweitens wollen wir nicht, dass Sie das Buch jetzt schon verzweifelt beiseitelegen.

Starten Sie einen X-Server auf Ihrem lokalen Linux-Rechner, läuft dieser i. d. R. unter der Nummer »0«. Dies wird in einer Umgebungsvariable verewigt, die den Namen DISPLAY\index{DISPLAY} trägt und den Wert :0 enthalten sollte[2]. Diese Information wird von den verschiedenen X-Clients ausgewertet und für den Zugriff auf den X-Server genutzt. Sollten Sie also mehrere X-Server auf Ihrem Rechner laufen lassen, weil Sie z. B. eine Dual-Head-Grafikkarte Ihr Eigen nennen und an dieser zwei Bildschirme angeschlossen sind, können Sie durch eine Veränderung dieser Umgebungsvariable die Clients auf die unterschiedlichen Monitore schicken[3].

Damit jetzt nicht Hinz und Kunz auf Ihrem X-Server Fenster ausgeben können, existiert noch ein weiterer Mechanismus, mit dem sich die X-Clients beim Server autorisieren müssen. Das geschieht heutzutage i. d. R. durch ein spezielles Cookie, das von einem Programm namens xauth erzeugt und verwaltet wird. Dieses wird in der Datei .Xauthority im Home-Verzeichnis Ihres Benutzerkontos abgelegt. Jedes X-Programm, das Sie unter Ihrer Kennung starten, wird aus dieser Datei das für den X-Server passende Cookie laden und sich damit autorisieren. Nur wenn dies mit dem vom X-Server verwalteten Wert übereinstimmt, wird eine grafische Ausgabe erlaubt.

```
victor@hugo:~$ echo $DISPLAY
:0
victor@hugo:~$ xauth list
hugo:0  MIT-MAGIC-COOKIE-1  61a0e9b797f9e5770865b2b4ae778544
victor@hugo:~$
```

Sie werden die Problematik sofort erfassen, wenn Sie eine Shell öffnen und dort das Kommando su - eingeben. Bei einem Aufruf dieser Form werden die Informationen über die Kennung, unter der Sie den X-Server gestartet haben, über Bord geworfen. Daher können ohne eine Anpassung der Umgebungsvariablen keine Programme auf der grafischen Oberfläche ausgegeben werden. Befindet man sich auf dem Rechner, auf dem die Datei mit den magischen Cookies liegt, kann man den X-Clients mit Hilfe der Umgebungsvariable XAUTHORITY eine Autorisierung ermöglichen:

```
victor@hugo:~$ su -
Password:
hugo:~# xclock
Error: Can't open display:
hugo:~# echo $DISPLAY
hugo:~# export DISPLAY=:0
hugo:~# xclock
Xlib: connection to ":0.0" refused by server
Xlib: No protocol specified
```

[2] Den Doppelpunkt kriegen wir später. :-)

[3] Das geht übrigens auch mit nur einer Grafikkarte. Sie können ja einmal auf eine Textkonsole umschalten und dort das Kommando X :1 aufrufen. Nett nicht? Sie erinnern sich auch noch daran, dass Sie mit STRG + ALT + BACKSPACE den X-Server wieder beenden können?

```
Error: Can't open display: :0
hugo:~# export XAUTHORITY=~victor/.Xauthority
hugo:~# xclock &
hugo:~#
```

Anstatt auf die Datei mit dem magischen Cookie zu verweisen, kann man dieses auch in der lokalen Xauthority-Datei des Nutzers abspeichern. Damit funktioniert der Zugriff dann ebenfalls. Wir demonstrieren Ihnen das noch einmal an einem lokalen Wechsel zur root-Kennung:

```
victor@hugo:~$ xauth list
hugo:0  MIT-MAGIC-COOKIE-1  61a0e9b797f9e5770865b2b4ae778544
victor@hugo:~$ su -
Password:
hugo:~# xauth list
xauth:  creating new authority file /root/.Xauthority
hugo:~# xauth add hugo:0  MIT-MAGIC-COOKIE-1  61a0e9b797f9e5770865b2b4ae778544
hugo:~# export DISPLAY=hugo:0
hugo:~# xclock &
hugo:~#
```

Praktischerweise hat die Ausgabe des Kommandos `xauth list` das Format, das Sie benötigen, um mit Hilfe von `xauth add ...` einen Eintrag einer Xauthority-Datei hinzuzufügen. Da bei jedem Start aber ein neues Cookie generiert wird, funktioniert diese Methode nur während der aktuell laufenden Sitzung.

Wir haben bisher immer lokal agiert, aber unser letztes Code-Beispiel gibt bereits einen kleinen Hinweis, wie wir X-Clients über das Netzwerk auf einen anderen Rechner schicken. Sie können vor dem Doppelpunkt in der `DISPLAY`-Variable einen Hostnamen oder eine IP-Adresse angeben. Wenn der X-Client dort eine Ausgabe erzeugen darf, wird das Fenster auf dem Monitor dieses Rechners erscheinen.

Achtung

Bei den meisten Distributionen ist der Aufruf des X-Servers mittlerweile so konfiguriert, dass das System keine Zugriffe über das Netzwerk annimmt. Dies hat keinerlei Auswirkung auf den Remote-Zugriff mit Hilfe von `ssh` und die darüber »verschickten« grafischen Ausgaben. Der einfache Netzzugriff wird darüber jedoch abgeklemmt. Davon, wie Sie Ihre Session starten, hängt es nun ab, an welcher Stelle Sie dem X-Server das Lauschen auf Netzwerkzugriffe erlauben wollen:

- **KDM und XDM**: Im Konfigurationsverzeichnis dieser Display-Manager finden Sie eine Datei `Xservers`, in der Aufrufe des X-Servers abgelegt sind inklusive der Kommandozeilen-Optionen. Hier

müssen Sie die Option `-nolisten tcp` entfernen, damit der X-Server auch auf Anfragen über das Netzwerk hört.

- **GDM**: In der Datei `gdm.conf` finden Sie einen Eintrag `DisallowTCP=True`. Diesen müssen Sie auf `False` setzen, damit der vom GDM gestartete X-Server auch Netzwerkzugriffe zulässt.
- **Auruf über die Kommandozeile**: Für den Start von X existiert ein Kommando mit dem Namen `startx`. Dieses nutzt die Funktionen von `xinit`, um eine vollwertige X-Session zu starten. Die Konfigurationsoptionen für den Aufruf des X-Servers finden sich in der Datei `xserverrc` des Konfigurationsverzeichnisses von `xinit`. Dort muss wie oben die Option `-nolisten tcp` entfernt werden, um einen direkten Netzwerkzugriff zu erlauben.

Bevor Sie den Netzwerkzugriff dauerhaft erlauben, sollten Sie sich aber noch unseren Tipp in Abschnitt 11.3.5 durchlesen. Die Sicherheitsmechanismen von X sind modernen Anforderungen nicht immer gewachsen.

Neben der Cookie-Verwaltung gibt es eine weitere Möglichkeit, Zugriffskontrollen auf den X-Server einzurichten und zu verwalten. Mit Hilfe des Kommandos `xhost` können Sie Regeln für Hosts festlegen und diesen einen Zugriff erlauben.

```
knoppix@knoppix:~$ /sbin/ifconfig eth0
eth0      Protokoll:Ethernet  Hardware Adresse 00:A1:B0:09:48:E2
          inet Adresse:192.168.1.11  Bcast:192.168.1.255  Maske:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5175 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6047 errors:0 dropped:0 overruns:1 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:2696235 (2.5 MiB)  TX bytes:804348 (785.4 KiB)
          Interrupt:18 Basisadresse:0xa400
knoppix@knoppix:~$ xhost
access control enabled, only authorized clients can connect
knoppix@knoppix:~$ xhost add +192.168.1.10
knoppix@knoppix:~$ xhost
access control enabled, only authorized clients can connect
INET:192.168.1.10
knoppix@knoppix:~$
```

Ein Aufruf des Kommandos `xhost` listet die zurzeit freigegebenen Rechner auf. Mit `xhost add +192.168.1.10` haben wir jetzt auf unserem Knoppix-Testrechner die IP-Nummer des Servers für den Zugriff freigegeben. Wichtig ist das Plus-Zeichen vor der Angabe des Rechners; damit geben Sie an, dass der Rechner Zugriff erhalten soll. Mit einem Minus-Zeichen können Sie eine Freigabe wieder zurückziehen. Jetzt können wir auf unserem Server die `DISPLAY`-Variable setzen und dann X-Programme auf dem Knoppix-System ausgeben.

```
victor@hugo:~$ export DISPLAY=192.168.1.11:0
victor@hugo:~$ xclock &
victor@hugo:~$
```

Zu diesem Vorgehen seien jedoch noch einige Bemerkungen erlaubt:

- Mit `xhost` erlauben Sie über die IP-Nummer bzw. den Rechnernamen *jedem* Nutzer dieses Computers den Zugriff auf den X-Server. Eine Differenzierung zwischen verschiedenen Kennungen ist nicht mehr möglich.
- Außerdem werden die Darstellungsinformationen für Fenster und deren Inhalte nicht geschützt, also verschlüsselt, übertragen. Für die Server-Wartung sind also die Möglichkeiten vorzuziehen, die `ssh` bietet. Durch die Tunnelung der X-Übertragung ist hierbei kein Abhören möglich.
- Trotzdem kann `xhost` ein nützliches Feature sein: Haben Sie z. B. ein X-Terminal, das Sie für ein einfaches Kiosk-System nutzen wollen, können Sie darauf den Server freigeben, von dem aus eine grafische Ausgabe erfolgen soll. Damit können Sie ohne großen Aufwand von einem entsprechend geschützten Server aus über das Netzwerk grafische Ausgaben erzeugen wie z. B. Monitorprogramme, die die aktuelle Systemauslastung darstellen, oder einen Webbrowser, der die aktuellen Busverbindungen ausgibt.
- `xhost` kann aber auch eine Falle sein: Sie sollten auf keinen Fall das Kommando `xhost +` ausführen! Schon gar nicht, wenn Ihnen die Kollegen weismachen wollen, dass dabei ja gar nichts passieren könnte, da Sie ja keinen Rechner angegeben haben. Probieren Sie es einmal heimlich alleine und ungesehen aus. Und erinnern Sie sich daran, dass es auch noch ein `xhost -` gibt!

3.1.1 Mit X-Terminals am Server anmelden

Eine verblüffend einfache Möglichkeit stellt der Zugriff eines X-Servers auf einen im Netzwerk befindlichen Display-Manager dar. Als Rechnerleistung noch hauptsächlich von großen Servern bereitgestellt wurde und der Zugriff i. d. R. über Terminals erfolgte, wurden auch für grafische Oberflächen geeignete Geräte gebaut, die sogenannten X-Terminals. Diese beinhalteten im Prinzip nur eine Grafikkarte, eine Netzwerkkarte (oder eine anders geartete Möglichkeit des Remote-Zugriffs) und die für den Netzwerkzugriff und die grafische Darstellung notwendige Software. Bei einer Session wurde die gesamte vom Nutzer gestartete Software auf dem Server ausgeführt. Nur die Ausgabe landete auf dem X-Terminal.

Die Nachfolger der X-Terminals sind die heute als Thin-Clients bezeichneten Geräte. Darunter wird so ziemlich alles subsumiert, was möglichst klein ist, wenig kostet und wenig tut. Wenn Sie mit alter Hardware gesegnet sind und überlegen, wie Sie mit wenig Geld möglichst viel erreichen können, ist diese antike Art des vernetzten Arbeitens vielleicht eine Option. In diesem Fall müssen Sie nur darüber nachdenken, ob die Fähigkeiten der alten PCs ausreichen, eine vernünftige grafische Ausgabe bezüglich Auflösung und Farbtiefe zu erzeugen. Im späteren Betrieb müssen die behäbig arbeitenden Prozessoren dann nur noch Pixel verwalten. Wenn auch die

Monitore noch vernünftig arbeiten, wäre nur die Anschaffung eines passend ausgestatteten Servers notwendig, weil dieser die gesamte Rechenarbeit übernehmen muss.

Wir wollen an dieser Stelle keine Anleitung geben, wie Sie aus einem PC ein X-Terminal machen. Hier gibt es bereits Distributionen, die Ihnen diese Aufgabe abnehmen. Je nach Hardware sollten Sie testen, welche mit dem geringsten Aufwand bei Ihnen einzusetzen sind. Da alle Distributionen das Booten über das Netzwerk bzw. über Diskette oder CD erlauben und eine Installation für einen ersten Test nicht notwendig ist, sollte der Test auch nicht unnötig Mühe machen.

Linux-Distributionen, die für X-Terminals geeignet sind

- Linux Terminal Server Project: `http://www.ltsp.org/`
- ThinStation: `http://thinstation.sourceforge.net/`
- ThinTUX: `http://thintux.sourceforge.net/`

Was wir Ihnen aber zeigen und erklären wollen, ist der Zugriff auf einen im Netzwerk laufenden Display-Manager mit Hilfe eines X-Servers. Dazu ist notwendig:

- ein Server oder irgendwie gearteter PC mit Linux und einem laufenden Display-Manager. Wir werden in unserem Beispiel KDM benutzen. Das Prinzip ist aber bei allen anderen Display-Managern gleich. Es muss übrigens kein Bildschirm am Rechner angeschlossen sein. Der Display-Manager kann auch ohne ein X auf Anforderungen warten.
- ein Client. Dieser sollte in unserem Fall im Textmodus laufen, da wir später den X-Server per Hand starten wollen. Wenn nicht, ist auch egal; Sie können das Ganze auch mit einem zweiten X-Server ausprobieren.
- ein Netzwerk, in dem sich beide Rechner befinden und das sich nicht gerade in einem Hochsicherheitsbereich befindet. Beide Rechner sollten sich »sehen« können, und am einfachsten ist es, wenn sich keine Hindernisse dazwischen befinden wie z. B. eine Firewall.

Der Informationsaustausch zwischen Display-Manager und X-Server findet über das Protokoll XDMCP, das **X** **D**isplay **M**anager **C**ontrol **P**rotocol statt. Dieses ist aus Sicherheitsgründen bei allen aktuellen Distributionen deaktiviert. KDM besitzt in der Konfigurationsdatei `kdmrc` (meistens ganz am Ende der Datei) eine Sektion `[Xdmcp]`. Dort muss der Eintrag `Enable` auf `True` gesetzt werden:

Listing 3.1: kdmrc

```
[Xdmcp]
Enable=True
Willing=/etc/kde3/kdm/Xwilling
```

Diejenigen, die KDM auf einem bildschirmlosen Server laufen lassen, können in der Datei `Xservers` (i. d. R. im selben Verzeichnis wie die Datei `kdmrc`) alle zu startenden X-Server auskommentieren.

Listing 3.2: Xservers

```
# Xservers - local X-server list
...

#:0 local@tty1 /usr/X11R6/bin/X -nolisten tcp :0
#:1 local@tty2 reserve /usr/X11R6/bin/X -nolisten tcp :1
#:2 local@tty3 reserve /usr/X11R6/bin/X -nolisten tcp :2
```

Man kann (und muss) dem Display-Manager mitteilen, welche Rechner berechtigt sind, seinen Service zu nutzen. Dazu existiert im gleichen Verzeichnis eine Datei `Xaccess`. Dort können wir für den Anfang die Zeile mit dem * auskommentieren, damit für die Testphase jede Anfrage angenommen wird:

Listing 3.3: Xaccess

```
# Xaccess - Access control file for XDMCP connections
#
...
*                          #any host can get a login window
...
```

Jetzt ist es an der Zeit, den Display-Manager zu starten (oder neu zu starten):

```
hugo:~# /etc/init.d/kdm restart
hugo:~#
```

Zusammengefasst müssen zwei Konfigurationsdateien angefasst werden:

1. Sie müssen XDMCP in der Datei `kdmrc` aktivieren, und
2. Sie müssen Host-Adressen in der Datei `Xaccess` eintragen, denen der Zugriff erlaubt ist.

Tipp

In der Datei `Xaccess` können Sie Hostnamen über Platzhalter angeben, z. B. mit »*.beispiel.org«. Die Platzhalter funktionieren nur leider nicht bei IP-Nummern.

Wenn der Zugriff trotzdem nicht funktioniert, müssen wir kontrollieren, ob wir uns eventuell durch Firewall-Einstellungen den Zugriff von außen abgeklemmt haben. Ein Blick in die Datei `/etc/services` zeigt uns, welche Ports für XDMCP und den Zugriff auf einen X-Server über das Netzwerk freigeschaltet sein müssen:

```
victor@hugo:~$ fgrep xdm /etc/services
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
victor@hugo:~$ fgrep x11 /etc/services
x11             6000/tcp        x11-0           # X Window System
x11             6000/udp        x11-0
x11-1           6001/tcp
x11-1           6001/udp
x11-2           6002/tcp
x11-2           6002/udp
x11-3           6003/tcp
x11-3           6003/udp
x11-4           6004/tcp
x11-4           6004/udp
x11-5           6005/tcp
x11-5           6005/udp
x11-6           6006/tcp
x11-6           6006/udp
x11-7           6007/tcp
x11-7           6007/udp
victor@hugo:~$
```

Das XDMCP-Protokoll nutzt die Ports 177, während die Netzwerk-Zugriffe auf einen X-Server über die Portnummer 6000 + Display-Nummer laufen. Wenn Sie es sich leisten können, schalten Sie für die Tests auf beiden Rechnern die Firewall ab; wenn nicht, müssen Sie die Portnummern für XDMCP und den von Ihnen gestarteten X-Server freigeben; bitte sowohl tcp als auch udp[4].

Damit wechseln wir auf den Client und versuchen dort einmal, uns ein Login von unserem Server zu besorgen. Wechseln Sie auf eine Textkonsole (wenn Sie nicht schon eine solche vor sich haben) und melden Sie sich dort mit einer normalen Kennung an. Alles am Netz? Kabel drin? Helm auf? X auf dem Client konfiguriert? Dann wollen wir mal:

```
victor@theodor:~$ X -broadcast

Fatal server error:
Server is already active for display 0
        If this server is no longer running, remove /tmp/.X0-lock
        and start again.

When reporting a problem related to a server crash, please send
the full server output, not just the last messages.
```

[4] Sie können das übrigens auch alles auf einem Rechner ausprobieren. Aber das wäre doch langweilig, oder?

```
Please report problems to submit@bugs.debian.org.
# Ach, Sie haben schon einen X-Server laufen?
victor@theodor:~$  X :1 -broadcast
```

Sollte bei ersten Aufruf eine Fehlermeldung erscheinen, versuchen Sie es mal mit dem zweiten. In diesem Fall läuft bereits ein X-Server auf Ihrem Client und wir starten den zweiten einfach auf einem anderen Display. Danach sollte die grafische Ausgabe starten und der Display-Manager des Servers auf Ihrem Monitor erscheinen. Jetzt können Sie sich mit einer Nutzer-Kennung anmelden, die auf dem Server gültig ist. Nach der Anmeldung wird dann auf dem Server die Desktop-Umgebung gestartet.

Next-Generation X: die NX-Clients und Server. Wenn Sie, egal ob mit Hilfe von SSH oder auf anderem Weg, den grafischen Remote-Zugriff auch einmal mit Netzwerkverbindungen mit geringer Bandbreite ausprobiert haben, werden Ihnen vor allem in puncto Geschwindigkeit die Nachteile dieser Art der Übertragung aufgefallen sein. Um insbesondere diese Problematik zu lösen ist, die Firma NoMachine mit ihren NX-Produkten angetreten. Teile der Software sind dabei unter der GPL veröffentlicht, sodass bereits auch mit *FreeNX* ein freier NX-Server existiert. NX-Clients werden von NoMachine kostenlos zur Verfügung gestellt.

3.2 Remote-Konfiguration und Administration mit SSH

Die Nutzung einer Kommandozeile auf Servern im Netzwerk ist so alt wie das Internet selbst: in den Anfängen erzählte man den Rechnern mit Hilfe von **Telnet** (Port 23) oder der **Remote Shell**, was sie zu tun und zu lassen hatten. Mit dem Wachstum des Internet und der Bedeutung der Server für Nutzer mit guter und vor allem wegen derjenigen mit böser Absicht wuchsen die Anforderungen an einen durch Kryptografie geschützten Zugriff, bei dem die übertragenen Daten nicht von dritter Seite belauscht werden konnten. Das Resultat **ssh** (Port 22) war neben den recht komplizierten Versuchen mit Kerberos ein einfach zu nutzender Client-Server-Ansatz als Ersatz für `telnet` bzw. `rsh`. Als die ssh-Entwickler ihr Produkt zu Geld machen wollten und in ein proprietäres Tool umwandelten, wurde eine freie Variante entwickelt, die – aufgespalten in die Projekte **openssl** und **openssh** – mittlerweile in allen Linux-Distributionen enthalten sind.

Secure Shell	Debian	Fedora	SUSE	Windows
Secure Shell-Client Die Client-Software, um über das Netzwerk auf andere Server zuzugreifen	ssh	openssh-clients	openssh	Putty OpenSSH f. Windows
Secure Shell-Server Die Server-Software, um Zugriff von außen zu ermöglichen	ssh	openssh-server	openssh	OpenSSH f. Windows

Tabelle 3.1: Paketübersicht

Secure Shell	Debian	Fedora	SUSE	Windows
SSH-Askpass Tool, das ssh die Nachfrage nach einem Passwort unter X ermöglicht	ssh-askpass ssh-askpass-gnome	openssh-askpass gaskpass	openssh-askpass openssh-askpass-gnome	—

Tabelle 3.1: Paketübersicht (Fortsetzung)

Wir werden Ihnen hier die grundlegenden Funktionen von SSH vorstellen. Man sollte die Secure Shell jedoch nicht unterschätzen: Hier sind mit wenigen Funktionen eine ganze Reihe von Einsatzmöglichkeiten vorhanden, die nicht auf den ersten Blick ersichtlich sind; dazu mehr in Abschnitt 11.2.3 auf S. 277.

3.2.1 Aktivierung des SSH-Servers

Da SSH zur Standardsoftware einer Linux-Installation gehört, müssen Sie sich i. d. R. nicht mehr um die Installation der entsprechenden Pakete kümmern. Während der Installation werden auch alle notwendigen Vorkehrungen getroffen, um den SSH-Server starten zu können, wie z. B. die Generierung der notwendigen kryptografischen Schlüssel. Der SSH-Server läuft als separater Prozess und wird nicht über den `inetd` (siehe auch Abschnitt 11.3.3) gestartet. Daher müssen Sie noch dafür sorgen, dass der Prozess beim Booten automatisch aufgerufen wird.

Debian	Am einfachsten ist es, mit Hilfe der `debconf`-Funktion für den automatischen Start zu sorgen. Rufen Sie diese mit `dpkg-reconfigure ssh` auf, und beantworten Sie die Frage, ob Sie den `sshd`-Server starten wollen mit »Ja«.
SUSE	Hier müssen Sie dafür sorgen, dass SSH in dem Runlevel eingetragen ist, in dem Sie den Server laufen lassen wollen. Wenn Sie Linux nicht explizit in einem anderen Runlevel starten, sollten Sie mit dem YaST-Modul *runlevel* dafür sorgen, dass SSH im 2. Runlevel gestartet wird.

Tabelle 3.2: Unterschiede zwischen den Distributionen

3.2.2 Der erste Remote-Zugriff mit SSH

Auch wenn Sie jetzt noch keinen Client zur Hand haben, von dem aus Sie einen SSH-Zugriff auf Ihren Server wagen können, lässt sich der Zugriff über die interne Loopback-Schnittstelle simulieren.

SSH arbeitet mit privaten und öffentlichen kryptografischen Schlüsseln: Die öffentlichen Schlüssel werden dabei von den Clients genutzt, um die Daten zu verschlüsseln, die zum Server übertragen werden. Der Server entschlüsselt die Daten dann mit Hilfe seines privaten Schlüssels. Der erste Schritt zu einer sicheren Verbindung ist also der Austausch der Schlüssel. In der Praxis schaut der SSH-Client in einer in Ihrem Home-Vereichnis gespeicherten Datei nach, ob dort passend zum privaten Schlüssel des Servers bereits ein öffentlicher Schlüssel gespeichert ist. Beim ersten

Aufruf ist dies nicht der Fall; daher lädt der Client den öffentlichen Teil des Schlüssels herunter und speichert ihn dort ab. Da dies ein sicherheitskritischer Schritt ist, fragt der Client nach, ob wir diesem Schlüssel Vertrauen entgegenbringen[5]. Danach wird das Passwort abgefragt: Wenn Sie nicht explizit eine andere Nutzerkennung angeben, versucht der SSH-Client sich auf dem »entfernten« Rechner mit der gleichen Kennung anzumelden, mit der Sie auf dem System angemeldet sind, von dem aus Sie den Zugriff versuchen.

```
victor@hugo:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 1c:9e:2c:91:59:50:7b:18:f9:05:5f:66:fd:27:b6:78.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost,127.0.0.1' (RSA) to the list of known hosts.
Password:# Passwort für Nutzer victor
# Die Begrüßungsmeldung des Servers
Last login: Fri Xxx 13 00:00:00 2005 from localhost
victor@hugo:~$ logout
victor@hugo:~$ cat .ssh/known_hosts
localhost ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEA3TBCZeggxiFoyXR5bm1cphChs6fcJEYviW//d7
qZjljqabnMOk8kL9lhDFLS2fsdO6urO2IFK1Vqmw5T12Ok7tvYeb9L2zQO4Ujn+puHUaqmMN9rsNg8o5d5Zj
KP9suYdHRzqi1SqAeA9jNlNzXbHA7xPOwAqj77mwF1pzGj/Ss=
victor@hugo:~$
```

Wenn die Autorisierung geklappt hat, erhalten Sie eine Shell, mit der Sie wie gewohnt arbeiten können. Außerdem wird beim ersten Aufruf in Ihrem Home-Verzeichnis ein Unterverzeichnis `.ssh` angelegt. Dort landen alle für die SSH-Nutzung mit dieser Kennung notwendigen Dateien, wie z. B. die Datei `known_hosts`, die die öffentlichen Schlüssel aller Server enthält, auf denen Sie sich bereits einmal eingewählt haben. Wir haben sie einmal mit `cat` ausgegeben. Die Einträge sind – wie hier jetzt nicht so genau zu erkennen – auf eine Zeile beschränkt: am Anfang steht der Name des Servers, dann der Typ des Schlüssels und danach der Schlüssel selbst. In dieser Datei können Sie auch »per Hand« Einträge ergänzen.

Sie können mit dem SSH-Client auch direkt ein Programm aufrufen:

```
victor@hugo:~$ ssh localhost date
Password:# Passwort für Nutzer victor
Fri Xxx 13 00:00:00 CET 2005
victor@hugo:~$
```

[5] Wir diskutieren an dieser Stelle nicht, welche Probleme hier auftreten könnten, werden aber später in Grundzügen darauf zurückkommen. Nur als Hinweis: eigentlich müsste der öffentliche Schlüsselteil auf einem anderen, sicheren Weg in die Datei `known_hosts` gelangen, da vor dem Austausch der Schlüssel die Verbindung noch nicht gegen Manipulationen geschützt ist. Theoretisch könnte jemand Ihnen mit einer »man-in-the-middle«-Attacke einen falschen Schlüssel unterschieben und danach die Verbindung zwischen Ihrem Client und dem Server belauschen.

In diesem Fall wird das Programm direkt ausgeführt und danach die Verbindung beendet. Da der öffentliche Schlüssel des Servers `localhost` jetzt bekannt ist, wird er nicht noch einmal abgefragt.

3.2.3 X-Forwarding mit SSH

Doch kommen wir zu den wirklich spannenden Dingen, mit denen wir Sie zum Staunen bringen wollen, vor allem, wenn Sie aus der Windows-Welt kommen. :-) Spätestens jetzt wäre es an der Zeit, einen zweiten Rechner mit einer Linux-Installation zur Hand zu haben, und sei es »nur« ein lauffähiges Knoppix. Wenn Sie Ihren Server mit `ping` finden, können wir Ihnen eine nützliche Funktion von SSH erklären, die eine nützliche Funktion von X noch nützlicher macht.

Zuerst müssen wir unseren SSH-Server aber noch dazu bewegen, das sogenannte X-Forwarding zuzulassen: damit ist gemeint, dass X-Programme, also Programme mit grafischer Ausgabe, ihr Fenster auf dem Client ausgeben können, während sie eigentlich auf dem Server ablaufen. Dazu muss jedoch auf der Client-Seite ein eigener X-Server laufen, weswegen wir diese Funktion am Beispiel einer zweiten Linux-Installation beschreiben wollen.

In der Konfigurationsdatei des `sshd` finden Sie gegen Ende einen Eintrag mit dem Namen `X11Forwarding`, der entweder auskommentiert oder auf »no« gesetzt ist. Setzen Sie diesen auf »yes«, und kontrollieren Sie, ob das `X11DisplayOffset` gesetzt ist.

Listing 3.4: /etc/ssh/sshd_config

```
...
X11Forwarding yes
X11DisplayOffset 10
...
```

Wenn Sie diese Änderungen gespeichert haben, müssen Sie den SSH-Server neu starten – wie immer am besten mit dem Init-Skript.

Wechseln wir nun die Konsole bzw. den Rechner. Im nächsten Beispiel wählen wir uns von einem mit Knoppix gebooteten Rechner in unseren Server ein:

```
knoppix@knoppix:~$ ssh -X victor@192.168.1.10
RSA key fingerprint is  1c:9e:2c:91:59:50:7b:18:f9:05:5f:66:fd:27:b6:78.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.10' (RSA) to the list of known hosts.
Password:# Passwort für Nutzer victor
# Die Begrüßungsmeldung des Servers
Last login: Fri Xxx 13 00:00:00 2005 from localhost
knoppix@knoppix:~$ xclock&
knoppix@knoppix:~$
```

Die Daten werden für die X-Clients transparent verschlüsselt übertragen. Versuchen Sie es einmal, und starten Sie z. B. `xclock`, ein einfaches X-Programm, das aber zu den Basispaketen von X gehört und auf jedem Rechner zu finden sein sollte.

Hinweis: Wenn's mal nicht so geklappt hat ...

Problem: Es kann keine Verbindung aufgebaut werden, und Sie erhalten als Fehlermeldung »`ssh: connect to host hugo port 22: Connection timed out`«. Oder der Server schließt die Verbindung direkt wieder: »`ssh_exchange_identification: Connection closed by remote host`«; oder die Verbindung wird zurückgewiesen: »`ssh: connect to host hugo port 22: Connection refused`«

Analyse: Das Gegenüber kann eine Verbindung im Prinzip auf zwei Ebenen ablehnen: Die erste Möglichkeit ist, dass eine Firewall-Einstellung dafür sorgt, dass die Pakete den SSH-Server gar nicht erst erreichen. In diesem Fall erhalten Sie das erste Symptom: Es werden nur Pakete an den Server gesendet, aber es kommen keine zurück, daher »Connection timed out«.

Die zweite Möglichkeit, bei denen Ihnen zumindest gesagt wird, dass Ihnen der Server die Türe vor der Nase zugeschlagen hat, deutet darauf hin, dass der SSH-Server bewußt Ihre Anfrage ablehnt. Ein Blick in die passende Log-Datei sollte dieses Verhalten belegen:

```
hugo:~# tail /var/log/auth.log
Xxx 13 00:00:00 hugo sshd[5880]: refused connect from 192.168.1.11
                                                        (192.168.1.11)
hugo:~#
```

In diesem Fall wurde dem SSH-Server mitgeteilt, dass entweder die Kennung oder der Rechner keine Erlaubnis hat, sich anzumelden. Diese Informationen können in der Konfigurationsdatei `sshd_config` oder über den `hosts_access`-Mechanismus, also Einträge in den Dateien `/etc/hosts.deny` und `/etc/hosts.allow` eingetragen worden sein.

»Connection refused« erhalten Sie in der Regel, wenn auf dem Port keine Anfragen beantwortet werden. Das kann bedeuten, dass der SSH-Server gar nicht läuft oder auf einer anderen Schnittstelle auf Anfragen wartet.

Lösung: Bei einem **Timeout** sollten Sie auf dem Zielserver für einen SSH-Versuch die Firewall deaktivieren bzw. direkt in der Firewall-Konfiguration dafür sorgen, dass SSH-Zugriffe erlaubt sind. In den anderen Fällen können Sie die Datei `/etc/hosts.allow` probeweise um eine Zeile `sshd: ALL` ergänzen. Sollte es danach funktionieren, müssten Sie nur noch das »ALL« durch auf Ihren Anwendungsfall passende IP-Bereiche ersetzen. Ansonsten hilft nur noch ein Blick in die Kon-

figurationsdatei des SSH-Dämons. Aber zuerst sollten Sie kontrollieren, ob er überhaupt läuft. ;-)

Problem: SSH beschwert sich beim Start der Verbindung mit der Meldung:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED        @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
04:07:77:c0:5a:6e:70:c2:1f:3e:4b:56:a2:b8:b3:38.
Please contact your system administrator.
Add correct host key in /home/victor/.ssh/known_hosts to get rid of this
message.
Offending key in /home/victor/.ssh/known_hosts:1
RSA host key for hugo has changed and you have requested strict checking.
Host key verification failed.
```

Analyse: Dies bedeutet, dass sich entweder das Schlüsselpaar geändert hat, mit dem der SSH-Server arbeitet oder dass unter seiner IP-Nummer in der Datei `.ssh/known_hosts` ein Schlüssel eines anderen Rechners aufgeführt ist. Solche Probleme treten häufig in Situationen auf, in denen Sie sich auf Rechnern einloggen, die per DHCP dynamisch eine IP-Nummer zugewiesen bekommen. Hier kann es passieren, dass ein Rechner eine IP-Nummer zugewiesen bekommt, die vorher einem anderen Rechner gehörte. Dieses Problem ist aber nicht von einer »man-in-the-middle attack« zu unterscheiden, daher die Warnung.

Lösung: Wenn Sie sich sicher sind, dass sich hier niemand zwischen Ihren Client und den Server schalten möchte, auf den Sie zugreifen wollen, reicht es aus, die jeweilige Zeile in der Datei `.ssh/known_hosts` zu löschen. In der Meldung ist übrigens ein Hinweis auf die problematische Zeile: `Offending key in /home/victor/.ssh/known_hosts:1`, also in der 1. Zeile der Datei `known_hosts`. Nach dem Löschen des Eintrags wird Ihnen beim nächsten Zugriff der Schlüssel des Servers erneut zum Download angeboten, und Sie haben wieder eine Verbindung.

3.3 Mit ganzen Desktops durch das Netz

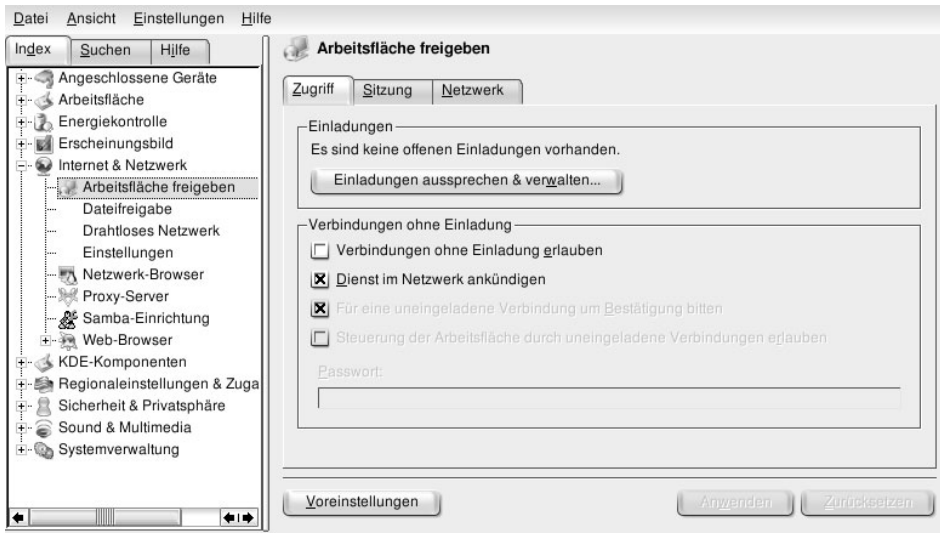
Auch wenn wir das Verschicken einzelner Fenster (und Programme) über das Netzwerk der Übertragung eines kompletten Desktops jederzeit vorziehen würden, seien hier doch die zurzeit wichtigsten Technologien erwähnt, die für den Remote-

Zugriff auf Desktops existieren. Mit der Software, die wir Ihnen jetzt vorstellen wollen, können Sie auch auf Windows-Rechner zugreifen. Falls Sie den Einsatz von Linux auf Arbeitsplatz-Rechnern planen, können Ihnen diese Tools helfen, Software zur Verfügung zu stellen, die z. B. nur für Windows oder ein anderes proprietäres Betriebssystem existiert.

Das Virtual Network Computing (VNC) ist wohl das am weitesten verbreitete Protokoll für einen Remote-Zugriff auf eine grafische Oberfläche, das sowohl als Client- als auch als Server-Lösung auf nahezu allen Plattformen implementiert ist. VNC hat jedoch den Nachteil, dass auf dem Zielrechner ein laufender Desktop existieren muss, der dann über das Netzwerk übertragen werden kann. Das bedeutet i. d. R., dass auf diesem Rechner ein Nutzer angemeldet sein und den Zugriff freigegeben haben muss. Damit eignet sich VNC z. B. gut für den Support, um Nutzern per Telefon und Remote-Zugriff bei Problemen zu helfen.

Wir wollen Ihnen am Beispiel von KDE zeigen, wie einfach die Freigabe eines Desktops und der Zugriff darauf ist. Dazu brauchen wir zwei Linux-Rechner, an denen sich jeweils ein Nutzer mit der grafischen Oberfläche KDE angemeldet hat. In unserem Szenario hat sich ein Kollege per Telefon gemeldet und benötigt Unterstützung bei einem Linux-Problem.

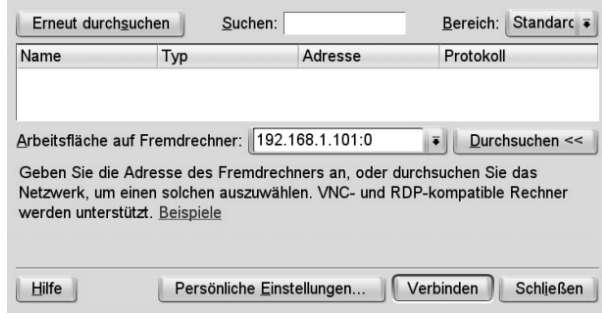
Zuerst muss der Kollege die Arbeitsfläche freigeben, damit wir von unserem Rechner aus darauf zugreifen können. Dies geht am einfachsten über das Kontrollzentrum von KDE. Dort findet er unter der Rubrik INTERNET & NETZWERK den Punkt ARBEITSFLÄCHE FREIGEBEN.



Dort wählt er den Button EINLADUNG AUSSPRECHEN & VERWALTEN und ruft im folgenden Dialog den Punkt NEUE PERSÖNLICHE EINLADUNG auf. Jetzt wird ein Server-Prozess gestartet, der auf einen Kontakt wartet. Zusätzlich ist ein Passwort

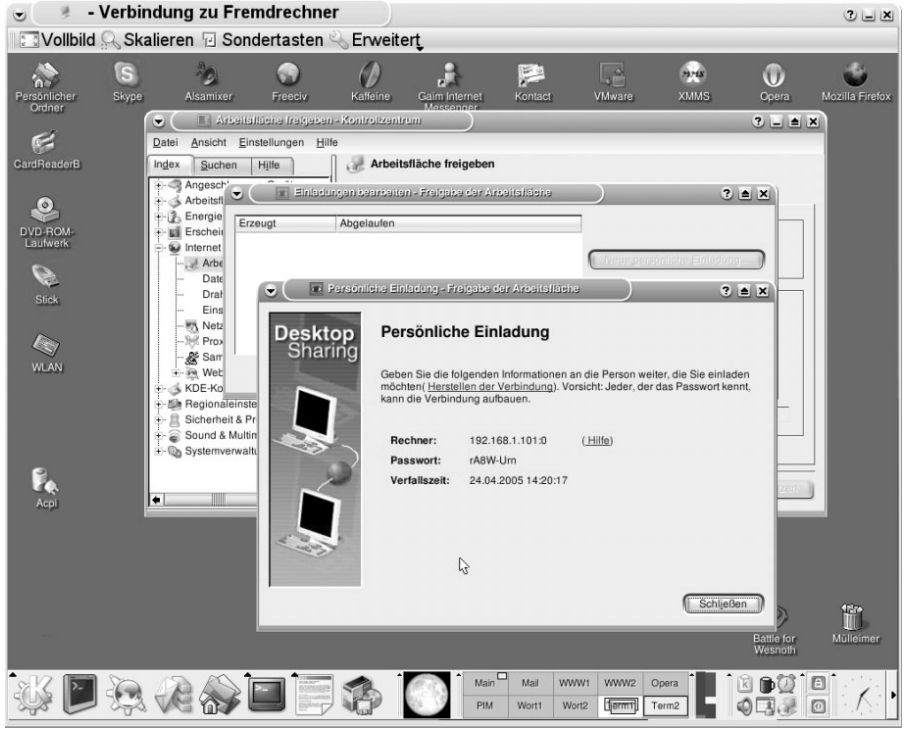
generiert worden, mit dem wir uns autorisieren müssen, um den Zugriff auf den Rechner zu erhalten.

Wir gehen einmal davon aus, dass der Kollege diese Aktionen mit unserer Hilfe über das Telefon hat ausführen können. Wir starten jetzt aus dem K-Menü unter der Rubrik INTERNET die VERBINDUNG ZU FREMDRECHNER; dahinter verbirgt sich die Software Krdc. Ein Versuch wäre, das Netzwerk nach der Freigabe zu durchsuchen. Wenn hier jedoch keine Rechner angezeigt werden, findet der Kollege am Telefon in seinem Freigabe-Dialog in der Zeile RECHNER die Informationen, die wir in das Feld FREMDRECHNER eintragen müssen.



Der nächste Dialog ermöglicht es uns, die Qualität der Verbindung festzulegen. Dies hat Auswirkungen darauf, welche Informationen wie über das Netzwerk übertragen werden, um ein flüssiges Arbeiten zu ermöglichen. Hier tun Sie sich keinen Gefallen, wenn Sie die Bandbreite überschätzen; bewegt sich die Maus später nur stockend und ruckend, sollten Sie die Verbindung noch einmal mit einer niedrigeren, angenommenen Bandbreite aufbauen. Jetzt wird der Kontakt hergestellt. Das Gegenüber muss jetzt den Zugriff durch uns bestätigen.

Danach werden wir nach dem Passwort gefragt, das für die Verbindung angelegt worden ist. Der Kollege am Telefon sollte bei der Mitteilung nicht vergessen, auf Groß- und Kleinschreibung hinzuweisen, da diese für die Anmeldung wichtig ist. Dann merken wir durch einem kleinen Fortschrittsdialog, dass versucht wird, die Verbindung herzustellen. Nach kurzer Zeit sollte dann ein Fenster erscheinen, in dem der Desktop des Kollegen angezeigt wird.



Innerhalb dieses Fensters wirken unsere Mausaktionen so, als würden wir vor diesem Rechner sitzen. Im Übrigen kann der Kollege alles mitverfolgen, was wir jetzt tun.

Für den Zugriff auf einen bildschirmlosen Server existiert mit TightVNC eine Software, die sich z. B. über eine SSH-Verbindung auf einem entfernten Rechner starten lässt. Danach kann dann über die Option VERBINDUNG ZU FREMDRECHNER eine VNC-Verbindung aufgebaut werden.

```
victor@hugo:~$ tightvncserver --help
TightVNC server version 1.2.9

Usage: tightvncserver [<OPTIONS>] [:<DISPLAY#>]
       tightvncserver -kill :<DISPLAY#>

<OPTIONS> are Xtightvnc options, or:

        -name <DESKTOP-NAME>
        -depth <DEPTH>
        -geometry <WIDTH>x<HEIGHT>
        -httpport number
        -basehttpport number
        -alwaysshared
        -nevershared
        -pixelformat rgb<NNN>
        -pixelformat bgr<NNN>

See tightvncserver and Xtightvnc manual pages for more information.
victor@hugo:~$ tightvncserver :5

You will require a password to access your desktops.

Password:
Verify:

New 'X' desktop is 192.168.1.10:5

Creating default startup script /home/victor/.vnc/xstartup
Starting applications specified in /home/victor/.vnc/xstartup
Log file is /home/victor/.vnc/hugo:5.log

victor@hugo:~$ tightvncserver -kill :5
Killing Xtightvnc process ID 19730
victor@hugo:~$ tightvncserver :5

New 'X' desktop is 192.168.1.10:5
```

```
Starting applications specified in /home/victor/.vnc/xstartup
Log file is /home/victor/.vnc/hugo:5.log

victor@hugo:~$
```

Der Start eines VNC-Servers ist dabei recht einfach. In der Regel genügt der Aufruf des Wrapper-Scripts `tightvncserver`. Alle wichtigen Angaben werden dann abgefragt und der Server unter der nächsten freien Display-Nummer gestartet. Es kann aber auch, wie wir es im obigen Beispiel getan haben, eine spezifische Display-Nummer angegeben werden. Zusätzlich wird im Konfigurationsverzeichnis unter anderem ein Script angelegt, das einen Window-Manager startet. Hier können Sie z. B. eintragen, das KDE oder vielleicht auch ein etwas ressourcensparenderer Desktop aufgebaut wird.

Der zweite Aufruf geht dann übrigens schneller, da TightVNC auf die beim ersten Aufruf gemachte Konfiguration zurückgreift. Sie beenden den VNC-Server mit dem Aufruf `tightvncserver -kill DISPLAYNR`; DISPLAYNR ergänzen Sie durch die Display-Nummer, mit der der Server gestartet wurde. Sie können auch mehrere VNC-Server unter verschiedenen Display-Nummern starten, falls dies notwendig sein sollte.

Den TightVNC-Server gibt es auch für Windows, jedoch bietet sich hier die Nutzung des RDP-Protokolls an, das die neueren Windows-Versionen von Haus aus mitbringen. Krdc bietet zusätzlich auch die Möglichkeit, eine RDP-Verbindung aufzubauen. Nur muss dazu i. d. R. noch rdesktop installiert werden; diese Software übernimmt die eigentliche Aufgabe des RDP-Zugriffs.

3.4 Auf der Höhe der Zeit mit (x)ntp

Ein Aspekt, den man oft vernachlässigt, dem aber bei der Administration einer wahrscheinlich immer größer werdenden Gruppe von Netzwerkrechnern eine wichtige Bedeutung zukommt, ist die Synchronisation der Systemzeit auf den verschiedenen Rechnern. Diese Aufgabe nehmen uns das **N**etwork **T**ime **P**rotocol und die nach ihm benannte Software (x)ntp (Port 123) ab.

Die Software bietet zwei Tools, die dazu dienen, die Systemzeit mit einem Zeitserver synchron zu halten: `ntpdate` setzt die Systemzeit, während der `ntpd` die Systemzeit aktuell hält. Was ist wann nötig? Wenn sich nicht gerade eine Funkuhr auf dem Motherboard des Rechners befindet, dessen Systemzeit aktuell gehalten werden muss, dann muss man leider davon ausgehen, dass die interne Rechneruhr mit der Zeit mehr oder weniger stark aus dem Ruder läuft. Dies geschieht aber in der Regel in Form berechenbarer Abweichungsintervalle. Der `ntpd` ermittelt durch die über einen längeren Zeitraum ablaufenden Synchronisationsprozesse Werte für diese Abweichung und kann dadurch die Systemzeit so genau halten, wie sie der Zeitserver liefert. Die Änderungen, die der NTP-Dämon an der Systemzeit vornimmt, liegen dabei in für das Funktionieren des Systems unkritischen Bereichen.

Wenn Sie die Möglichkeiten des NTP nutzen wollen, müssen Sie nicht unbedingt selbst einen Server betreiben, der an eine Atomuhr angeschlossen ist. Es gibt eine Reihe von Servern im Netzwerk, die als Zeitserver fungieren können. Das Projekt NTP, das die Software programmiert hat, führt eine Liste der öffentlichen NTP-Server. Außerdem stellt es mit den `pool`-Adressen einen Service zur Verfügung, der eine gleichmäßige Auslastung der öffentlichen Server gewährleistet. Wenn Sie die Adresse `de.pool.ntp.org` benutzen, wird Ihnen automatisch ein Zeitserver zugewiesen. Wenn Sie diese Adresse z. B. mehrfach in der Konfigurationsdatei für den `ntpd` eintragen, wird dieser sich mit verschiedenen Servern synchronisieren. Damit ist gewährleistet, dass für diejenigen, die diesen Service kostenfrei zur Verfügung stellen, kein unnötiger Netzwerkverkehr entsteht.

Beginnen wir jedoch von vorn: zuerst sollten Sie die Uhrzeit des Rechners auf einen möglichst aktuellen Wert stellen. Wenn der Rechner bereits ans Internet angeschlossen ist, können Sie diese Aufgabe dem Programm `ntpdate` überlassen.

Achtung

Wir möchten an dieser Stelle zur Vorsicht raten! Achten Sie darauf, dass die Uhrzeit des Rechners, auf dem Sie die Systemzeit aktualisieren wollen, bereits möglichst nahe an der realen Zeit liegt. Bei einem größeren Zeitsprung könnten die laufenden Prozesse sonst auf »dumme Gedanken« kommen! Ein plötzlich anspringender Bildschirmschoner ist dabei noch das geringste Übel …

Ein Blick auf `date` gibt uns die aktuelle Systemzeit aus. Diese liegt – wie ein Vergleich mit der Armbanduhr oder sonst einem erreichbaren Chronometer zeigt – im Bereich der aktuellen Zeitrechnung. Jetzt können wir mit `ntpdate` die Zeit genauer stellen lassen:

```
hugo:~# date
Mon xxx 16 09:06:41 CEST 2005
hugo:~# ntpdate de.pool.ntp.org
16 xxx 09:07:17 ntpdate[6487]: step time server 84.16.227.162 offset 26.936620 sec
hugo:~# ntpdate de.pool.ntp.org
16 xxx 09:07:22 ntpdate[6490]: adjust time server 84.16.227.162 offset -0.000293 sec
hugo:~#
```

Sie erkennen beim zweiten Aufruf, dass der Offset zum Zeitserver nur noch im Bereich von zehntausendstel Sekunden liegt. Mit dieser Genauigkeit lässt sich jetzt der Zeitserver starten:

```
hugo:~# /etc/init.d/ntp-server start
hugo:~# ntpq -p
```

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 mabuse.homeip.n 192.53.103.104   2 u    5   64    1   39.181   10.142   0.004
 binky.tuxfriend 192.53.103.104   2 u    4   64    1   29.935   x8.660   0.004
 anton.hin.de    130.149.17.21    2 u    3   64    1   24.276   11.959   0.004
 LOCAL(0)        LOCAL(0)        13 l    2   64    1    0.000    0.000   0.004
hugo:~#
```

Wir haben in der Konfigurationsdatei dreimal einen Eintrag `de.pool.ntp.org` gemacht: Während `ntpdate` zur besseren Ermittlung des Offsets den gleichen Server zugewiesen bekommt, erhält der `ntpd` drei verschiedene Server, mit denen er sich synchronisieren kann.

NTP für das lokale Netzwerk konfigurieren: Wir haben die Konfigurationsdatei jetzt schon zweimal erwähnt, nun wollen wir Ihnen die wichtigen Einträge kurz vorstellen: Wenn Sie nur einen Client einrichten, müssen Sie in der Konfigurationsdatei nur einen oder mehrere Server angeben, mit denen die Zeit synchronisiert werden soll. Hier sehen Sie die Einträge, die wir für das obige Beispiel genutzt haben.

Listing 3.5: /etc/ntp.conf

```
...
# pool.ntp.org maps to more than 100 low-stratum NTP servers.
# Your server will pick a different set every time it starts up.
#  *** Please consider joining the pool! ***
#  ***  <http://www.pool.ntp.org/#join>  ***
#server pool.ntp.org
server de.pool.ntp.org
server de.pool.ntp.org
server de.pool.ntp.org
```

Wenn Sie im lokalen Netzwerk einen Zeitservice verwenden wollen, sollten Sie jedoch nicht alle Rechner mit externen Servern synchronisieren. Es reicht, zwei oder drei Rechner dafür einzurichten. Diese können dann als lokale Zeitserver fungieren. Damit diese aber auch als Server von den Clients akzeptiert werden, sollten Sie ihnen einen höheren `stratum`-Wert geben, als in der Konfigurationsdatei als Standard eingetragen ist.

Listing 3.6: /etc/ntp.conf

```
# ... and use the local system clock as a reference if all else fails
# NOTE: in a local network, set the local stratum of *one* stable server
# to 10; otherwise your clocks will drift apart if you lose connectivity.
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Das Stratum dient dem Netzwerk der Zeitserver als Identifikation der Wichtigkeit bzw. Nähe eines Servers zu einer Zeitquelle, z. B. einer Funkuhr. Die Rechner, die

direkt an eine solche Quelle angeschlossen sind, erhalten das Stratum 0; diejenigen, die diese zum synchronisieren nutzen, das Stratum 2 usw. Ab einem gewissen Stratum-Wert nehmen die NTP-Dämonen die Zeitinformationen eines Servers nicht mehr »ernst« und verlassen sich lieber auf die eigene Hardware-Uhr, auch wenn die Zeit des genutzten Servers genauer ist.

Automatische Uhren im lokalen Netz: Sie können sich die Konfiguration der NTP-Clients vereinfachen, wenn Sie die Zeitserver ihre Uhrzeit per Broadcast propagieren lassen. Dazu ergänzen Sie auf den Servern in der Konfiguration eine `restrict`- und eine `broadcast`-Zeile:

Listing 3.7: /etc/ntp.conf

```
...
# Local users may interrogate the ntp server more closely.
restrict 127.0.0.1 nomodify
restrict 192.168.1.0 mask 255.255.255.0 nomodify

# If you want to provide time to your local subnet, change the next line.
broadcast 192.168.1.255
...
```

Die Clients müssen jetzt noch darauf geeicht werden, sich die Informationen über die Zeit per Broadcast zu suchen. Dazu kommentieren Sie die letzten beiden Zeilen in der Konfigurationsdatei aus:

Listing 3.8: /etc/ntp.conf

```
# If you want to listen to time broadcasts on your local subnet,
# de-comment the next lines. Please do this only if you trust everybody
# on the network!
disable auth
broadcastclient
```

Das von uns hier beschriebene Beispiel arbeitet ohne Authentisierung. Nehmen Sie die Warnung in der Konfigurationsdatei daher ernst. Informationen zu Authentisierung und weitere wichtige Informationen finden Sie in den unten angegebenen Literaturhinweisen.

Zum Schluss noch ein Beispiel für eine etwas langsamere Synchronisation der Systemzeit: der von uns installierte Rechner ging ca. 15 Minuten nach. Anstatt die Systemzeit mit `ntpdate` zu setzen, haben wir den NTP-Dämon gestartet und eine Weile laufen lassen.

Listing 3.9: /var/log/daemon.log

```
Xxx 29 18:25:15 theodor ntpd[6243]: ntpd 4.2.0a@1:4.2.0a+stable-8-r Sat Mar 19 14:14:09
                                                                              CET 2005 (1)
Xxx 29 18:25:15 theodor ntpd[6243]: precision = 3.000 usec
```

```
Xxx 29 18:25:15 theodor ntpd[6243]: Listening on interface wildcard, 0.0.0.0#123
Xxx 29 18:25:15 theodor ntpd[6243]: Listening on interface wildcard, ::#123
Xxx 29 18:25:15 theodor ntpd[6243]: Listening on interface lo, 127.0.0.1#123
Xxx 29 18:25:15 theodor ntpd[6243]: Listening on interface eth0, 192.168.1.10#123
Xxx 29 18:25:15 theodor ntpd[6243]: kernel time sync status 0040
Xxx 29 18:28:32 theodor ntpd[6243]: synchronized to LOCAL(0), stratum 13
Xxx 29 18:29:34 theodor ntpd[6243]: synchronized to 193.218.127.251, stratum 2
Xxx 29 18:59:55 theodor ntpd[6243]: time reset +856.891493 s
```

Der `ntpd` hat in diesem Beispiel eine ganze Weile gewartet und dann die Uhrzeit so vorgestellt, dass bei der Angleichung keine Stundengrenze überschritten wurde.

Weitere Literatur

- `www.ntp.org`
- »TimePrecision-HOWTO«
- `man ntpd`
- `man ntpdate`

3.5 Einrichtung eines einfachen DNS-Proxys/Servers

Jetzt haben Sie den ersten Server und vielleicht auch schon den ersten Client eingerichtet; da liegt es nahe, den Kindern Namen zu geben und die Rechner auch darüber anzusprechen. Dazu benötigt man den sogenannten Nameservice bzw. Domain Name ServiceDomain Name System, kurz DNS (siehe auch Abschnitt 2.1.2 auf S. 23). Dieser läuft als Server-Prozess auf einem Rechner im Netzwerk (häufig dem Gateway), mit dessen Hilfe die Clients die Namen in IP-Nummern auflösen können. Standard für diesen Dienst ist der BIND-Server, der **B**erkley **I**nternet **N**ame **D**omain Server, der sich rühmen kann, die meistgenutzte DNS-Server-Software im Internet zu sein. Andererseits ist er aber auch ein Monstrum und die Pflege eines DNS-Servers ist für einen Anfänger eine Strafe. Daher wollen wir Ihnen für Ihr lokales Netzwerk eine andere Lösung anbieten, die Ihnen helfen wird, einen internen Domain Name Service aufzusetzen. Wenn Sie jedoch eigene Domains selbst verwalten wollen, kommen Sie um BIND nicht herum.

Was tut DNSMasq für Sie?

- Sie können in der Datei `/etc/hosts` auf einfache Weise Namen und IP-Nummern für die Rechner in Ihrem lokalen Netzwerk vorhalten. DNSMasq liest diese Datei und liefert sie auf Anfrage aus.
- DNSMasq kann als Proxy für Ihr lokales Netzwerk arbeiten. Das heißt, dass alle DNS-Anfragen über einen Server abgewickelt werden können, der diese zwischenspeichert. Dadurch wird die Außenverbindung entlastet, und die Namensauflösung wird deutlich beschleunigt.
- Zusätzlich bietet DNSMasq noch einfache DHCP-Funktionen an, die wir in Kapitel 5 ab S. 89 jedoch ausführlicher anhand der DHCP-Software des Internet System Consortium erläutern.

dnsmasq	Debian	Fedora	SUSE
dnsmasq	dnsmasq	dnsmasq	dnsmasq
nslookup Tools zum Test des DNS-Systems	dnsutils	bind-utils	bind-utils

Tabelle 3.3: Paketübersicht

Nach der Installation finden Sie im Verzeichnis `/etc` die Konfigurationsdatei `dnsmasq.conf`. Diese muss je nach Anwendungsfall noch ein wenig angepasst werden. Betrachten wir dazu zwei Fälle:

Fall 1: Server mit statischer IP-Nummer

Im einfachen Fall wollen Sie einen Ihrer Rechner im lokalen Netzwerk, bei dem die Netzwerkanbindung bereits funktioniert, um die DNS-Proxy-Funktion erweitern. In diesem Fall brauchen Sie an der Standard-Konfiguration eigentlich nichts zu ändern! Wie arbeitet DNSMasq in diesem Fall?

- Die Informationen, welchen DNS-Server DNSMasq nach Adressen fragen soll, werden der Datei `/etc/resolv.conf` entnommen.
- DNSMasq stellt den Service automatisch auf jeder Netzwerkschnittstelle des Rechners zur Verfügung. Die Anfragen werden auf dem Port 53 beantwortet, dem Standard für den Domain Name Service.

Wenn Sie also DNSMasq nach der Installation über das Init-Script starten, sollten eine einfache Anfrage bereits funktionieren:

```
hugo:~# /etc/init.d/dnsmasq start
Starting DNS forwarder and DHCP server: dnsmasq.
hugo:~# nslookup www.uni-duesseldorf.de localhost
Server:         localhost
Address:        127.0.0.1#53

Name:   www.uni-duesseldorf.de
Address: 134.99.128.40
hugo:~#
```

Zum Testen nutzen wir das Tool `nslookup`, das i. d. R. auf Ihrem System bereits installiert ist. Sie können es mit einer oder mit zwei Optionen aufrufen: Wenn Sie testen möchten, ob die lokale DNS-Konfiguration funktioniert, rufen Sie `nslookup` nur mit einem Rechnernamen auf, der aufgelöst werden soll; wenn Sie die Funktion eines im Netz befindlichen DNS-Servers testen wollen, geben Sie `nslookup` noch die Adresse dieses Servers mit, damit es dort versucht, den Namen aufzulösen.

Jetzt wollen wir einen »privaten« Rechnernamen definieren, den wir im lokalen Netzwerk nutzen wollen. Dazu ergänzen wir die Datei `/etc/hosts` um eine Zeile für unseren Client:

Listing 3.10: /etc/hosts

```
192.168.1.11  theodor
192.168.1.12  antonius
```

Nach einem Neustart des Server-Prozesses liefert uns DNSMasq auch die IP-Nummern dieses Rechners auf Anfrage zurück. Dass dies vom Rechner `hugo` aus funktioniert, glauben Sie uns wahrscheinlich. Dazu wäre DNSMasq im Prinzip auch nicht notwendig gewesen, da von den dort laufenden Programmen die Datei `/etc/hosts` direkt ausgewertet wird. Loggen Sie sich daher einmal auf einem Ihrer Linux-Clients ein und versuchen Sie von dort aus, eine DNS-Anfrage beantwortet zu bekommen:

```
victor@theodor:~$ nslookup antonius 192.168.1.10
Server:         192.168.1.10
Address:        192.168.1.10#53

Name:   antonius
Address: 192.168.1.12
victor@theodor:~$
```

Dieses Ergebnis ist schon interessanter: Wir haben über die IP-Nummer unseres Servers nach einem Hostnamen gefragt und eine Antwort bekommen. Jetzt könnten wir die Datei `/etc/resolv.conf` dahingehend ändern, dass wir dort diesen Server als DNS-Server eintragen:

Listing 3.11: /etc/resolv.conf

```
nameserver 192.168.1.10
```

Ab jetzt werden von diesem Client aus alle Anfragen über unseren DNS-Proxy abgewickelt, inklusive der dort in der Datei `/etc/hosts` gespeicherten Namen. Dadurch können Sie jetzt von theodor auf antonius über dessen Namen zugreifen.

Jetzt können Sie auf allen Clients die 192.168.1.10 als DNS-Server eintragen, nur auf dem Server selbst wird noch direkt auf die »Originale« zugegriffen. Dies liegt an den Einträgen in der `/etc/resolv.conf`. Würden wir hier ohne zusätzliche Maßnahmen die 127.0.0.1 eintragen, würden alle lokalen Programme sich an DNSMasq wenden, aber DNSMasq selber leider auch. Glücklicherweise existiert eine Möglichkeit, dieses Henne-Ei-Problem zu umgehen: Sie können die Server, an die DNSMasq die Anfragen weiterleiten soll, in der Konfigurationsdatei angeben:

Listing 3.12: /etc/dnsmasq.conf

```
[...]
# If you don't want dnsmasq to read /etc/resolv.conf or any other
# file, getting its servers from this file instead (see below), then
# uncomment this
```

```
no-resolv

# If you don't want dnsmasq to poll /etc/resolv.conf or other resolv
# files for changes and re-read them then uncomment this.
no-poll

# Add other name servers here, with domain specs if they are for
# non-public domains.
server=nnn.nnn.nnn.nnn # Ein erster Nameserver
server=nnn.nnn.nnn.nnn # Ein zweiter Nameserver
[...]
```

Nach einem Neustart des DNSMasq-Servers können Sie jetzt die Datei `/etc/resolv.conf` auf eine Zeile der Art `nameserver 127.0.0.1` reduzieren. Dadurch werden alle lokalen Anfragen auch über DNSMasq abgewickelt.

Fall 2: Server als Gateway für die Einwahl ins Internet

Wenn Sie sich mit Ihrem Rechner über Modem, ISDN oder DSL ins Internet einwählen, ist der Einsatz von DNSMasq besonders wünschenswert, da die Netzwerkverbindung ja nicht auch noch durch häufige DNS-Anfragen belastet werden muss. Hier haben wir aber i. d. R. das Problem, dass bei der Einwahl die IP-Nummer dynamisch vergeben wird. Was ist zu tun?

- Nach »Außen« bei der Weiterleitung der Anfragen ergeben sich Probleme, weil wir DNSMasq bei jeder Einwahl die aktuell gültigen DNS-Server mitteilen müssen, insbesondere dann, wenn der Gateway-Rechner selbst auch über DNSMasq DNS-Anfragen absetzen soll.
- Nach »Innen« bleibt alles beim Alten; es sollte nur dafür gesorgt werden, dass DNSMasq nicht auf der Einwahl-Schnittstelle DNS-Anfragen beantwortet.

Warum gibt es bei der Einwahl Probleme? Da jeder Provider i. d. R. eigene Nameserver besitzt und damit sich je nach Einwahl auch deren IP-Nummern ändern, verändert der für die Einwahl zuständige Dämon die Datei `/etc/resolv.conf`, um sie den aktuellen Gegebenheiten anzupassen. Das würde aber unseren Eintrag `nameserver 127.0.0.1` überschreiben. Also müssen wir DNSMasq auf andere Weise mitteilen, wo die aktuell gültigen Informationen bezüglich der zu nutzenden DNS-Server stehen.

Der PPP-Dämon, der für die Einwahl über Modem und DSL genutzt wird (und der `ipppd` für die Einwahl per ISDN auch) kann so konfiguriert werden, dass er eine separate `resolv.conf` anlegt und darin die aktuellen DNS-Server speichert. Dies wird mit Hilfe der Option `usepeerdns` angeschaltet, die Sie in der Datei `/etc/ppp/options` eintragen können. Danach müssen Sie nur noch DNSMasq mitteilen, wo die auszuwertende `resolv.conf` zu finden ist:

Listing 3.13: /etc/dnsmasq.conf

```
[...]
# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
resolv-file=/etc/ppp/resolv.conf

# By  default,  dnsmasq  will  send queries to any of the upstream
# servers it knows about and tries to favour servers to are  known
# to  be  up.  Uncommenting this forces dnsmasq to try each query
# with  each  server  strictly  in  the  order  they   appear   in
# /etc/resolv.conf
#strict-order

# If you don't want dnsmasq to read /etc/resolv.conf or any other
# file, getting its servers from this file instead (see below), then
# uncomment this
#no-resolv

# If you don't want dnsmasq to poll /etc/resolv.conf or other resolv
# files for changes and re-read them then uncomment this.
#no-poll
[...]
```

Sie sollten darauf achten, dass die Optionen `no-resolv` und `no-poll` in diesem Fall *nicht* aktiviert sind!

Debian-Tipp

Die Debian-Entwickler haben diese Probleme mit einem eigenen Tool für die Pflege der `/etc/resolv.conf` umgangen. Wenn Sie DNSMasq zusammen mit dem Paket `resolvconf` installieren, werden alle oben beschriebenen Schritte automatisch erledigt. Außerdem ist die Software PPP-Dämon und alle weiteren, die die `/etc/resolv.conf` manipulieren, ebenfalls an das Paket `resolvconf` angepasst worden. Das geht soweit, dass bei einer Änderung der DNS-Konfiguration alle Prozesse, die diese Informationen nutzen, darüber informiert respektive neu gestartet werden, wenn das notwendig ist.

Um DNSMasq dazu zu überreden, die Einwahl-Schnittstelle nicht zu bedienen, haben Sie zwei Möglichkeiten: Sie können entweder die Schnittstelle ausschließen, oder explizit die IP-Nummern angeben, auf denen DNSMasq Anfragen beantworten soll:

- Im ersten Fall sorgen Sie mit der Option `except-interface=ppp0` z. B. dafür, dass DNSMasq das Interface `ppp0` ignoriert, das bei der Einwahl als ausgehende Netzwerkschnittstelle angelegt wird.

- Im zweiten Fall geben Sie mit der Option `listen-address=192.168.1.10` z. B. an, dass DNSMasq nur an der Schnittstelle antworten soll, die mit dieser IP-Nummer konfiguriert ist. Vergessen Sie in diesem Fall nicht, eine weitere Zeile `listen-address=127.0.0.1` hinzuzufügen, damit auch lokale Anfragen verarbeitet werden!

Je nach Einsatzszenario kann das eine oder andere Verfahren sinnvoller sein. Diese Entscheidung wollen wir aber Ihnen überlassen. Weitere Hinweise zu DNSMasq finden Sie im Dokumentationsverzeichnis, das mit dem Paket installiert wird oder auf der Webseite des Projekts.

Weitere Literatur

- Homepage des Projekts: `http://thekelleys.org.uk/dnsmasq/doc.html`
- `/usr/share/doc/dnsmasq/setup.html`