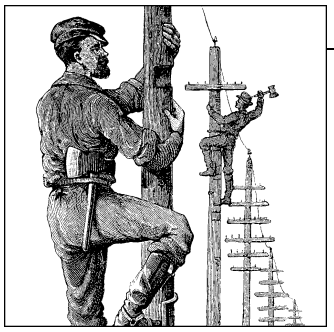

KAPITEL 10

Das Netzwerk einrichten

In diesem Kapitel werden wir die Hardware konfigurieren, die wir benötigen, um sowohl mit unserem lokalen Netz als auch dem Internet in Verbindung zu treten. Die Beschreibung habe ich dabei so gehalten, daß weniger die Werkzeuge einer bestimmten Distribution beschrieben werden als vielmehr ein Vorgehen, das mit jeder Distribution funktionieren sollte.

Vorbereitung

Bevor wir die Netzwerk-Interfaces einrichten, sollten wir alle Kabel aus den Netzwerk- und Telefondosen entfernen. Ohne diese Vorsichtsmaßnahme riskieren wir, das System in einen Zustand zu bringen, in dem der Rechner schon mit dem Internet verbunden ist, er aber noch nicht durch das Firewalling geschützt ist. Obwohl wir schon die meisten Einfalltore in den Rechner eliminiert haben, sollten wir kein Risiko eingehen.

Wir werden daher im folgenden die Netzwerkhardware konfigurieren und uns dann überlegen, wie wir sie testen, ohne den Rechner einer Gefahr auszusetzen. Danach wissen wir, daß die Devices (Netzwerkkarte, ISDN-Adapter und Modems) funktionieren, und können mit anderen Konfigurationen weitermachen, ohne uns später beim Auftreten eines Fehlers fragen zu müssen, ob das Problem in der Hardwarekonfiguration oder doch z. B. in den Paketfilterregeln liegt.

Konfiguration der Netzwerkkarte

Beginnen wir als erstes mit der Netzwerkkarte. Sie werden in Ihrer Firewall mindestens eine brauchen, um den Kontakt zum lokalen Netz herzustellen. Gehören Sie zu den Glücklichen, die über eine Standleitung an das Internet angebunden sind, so hat Ihnen Ihr Provider wahrscheinlich einen Router zur Verfügung gestellt, der die eigentliche Anbindung übernimmt. In so einem Fall werden Sie auch die Verbindung zum Internet über eine Netzwerkkarte herstellen.

Um eine Netzwerkkarte zu administrieren, benutzt man den Befehl `ifconfig`. So kann man sich mit dem Aufruf

```
# ifconfig -a
```

die Eigenschaften der vorhandenen Netzwerk-Interfaces anzeigen lassen. Ohne den Parameter `-a` werden dagegen nur diejenigen Netzwerk-Interfaces angezeigt, die schon konfiguriert wurden.

Um ein Netzwerk-Interface zu konfigurieren und in Betrieb zu nehmen, genügt es, `ifconfig` mit den Parametern

```
# ifconfig <Interf.> <Adresse> netmask <Maske> \
> broadcast <Broadcast-Adresse> up
```

aufzurufen. Haben wir z. B. nur eine Netzwerkkarte installiert, die die Adresse 192.168.0.99, die Netzmaske 255.255.255.0 und die Broadcast-Adresse 192.168.0.255 erhalten soll, so lautet der Aufruf:

```
# ifconfig eth0 192.168.0.99 netmask 255.255.255.0 \
> broadcast 192.168.0.255 up
```

Weitere Interfaces werden auf dieselbe Art konfiguriert. Die erste vom Kernel erkannte Netzwerkkarte hat dabei die Bezeichnung `eth0`, das zweite Interface `eth1` etc.

Um nun eine Netzwerkkarte zu deaktivieren, benutzt man den Befehl

```
# ifconfig <Interface> down
```

Im beschriebenen Beispiel lautet der Aufruf also

```
# ifconfig eth0 down
```

Schließlich kann man sich mit

```
# ifconfig eth0
```

die aktuelle Konfiguration eines Interfaces anzeigen lassen.

Ein zusätzlicher Faktor bei der Netzwerkkonfiguration ist das Routing. Für jedes zu sendende Paket muß der Kernel entscheiden, über welches Interface es geschickt werden soll und ob es direkt zugestellt werden kann oder ob es an einen Gateway-Rechner gesendet werden muß, welcher es dann in entfernte Netze weiterleitet. Hierzu verwaltet der Kernel eine Tabelle mit Einträgen der Art

<Zieladresse> <Gateway-Adresse> <Netzmaske> <Interface>

Als wir im obigen Beispiel das Interface `eth0` konfiguriert haben, wurde automatisch ein Eintrag in dieser Tabelle erzeugt, wie der folgende Aufruf zeigt:

```
# route -n
Kernel IP routing table
Destination  Gateway  Genmask        Flags Metric Ref Use Iface
192.168.0.0  0.0.0.0  255.255.255.0  U     0      0     0 eth0
127.0.0.0    0.0.0.0  255.0.0.0      U     0      0     0 lo
```

Die Felder »Flags«, »Metric«, »Ref« und »Use« wollen wir hier nicht näher betrachten. Bei Bedarf liefert der Aufruf von »`man route`« weitere Erklärungen. Den Parameter »-n« benutzen wir, damit der Rechner nicht versucht, logische Namen für die IP-Adressen zu finden. Bevor wir die Namensauflösung nicht konfiguriert haben, macht ein derartiger Versuch keinen Sinn, sondern kostet nur Zeit. Das Interface `lo` ist schließlich immer vorhanden und kann daher von uns getrost ignoriert werden.

Der Eintrag für Interface `eth0` in unserem Beispiel besagt, daß alle Adressen im Bereich von 192.168.0.0 bis 192.168.0.255 über das Interface `eth0` versendet und direkt zugestellt werden sollen. Dies reicht völlig, wenn das Interface mit einem lokalen Netz verbunden ist, in dem alle Rechner direkt erreichbar sind.

Anders liegt der Fall, wenn im lokalen Netz Gateways existieren, die den Kontakt zu anderen Netzen herstellen. Hier müssen wir dem Kernel sowohl mitteilen, daß diese anderen Netze existieren, als auch über welchen Rechner er Pakete an besagte Netze schicken kann. Hierzu ist ein zusätzlicher Eintrag in der Routing-Tabelle nötig. Dies kann mit dem Aufruf

```
# route add -net <Zieladresse> gw <Gatewayadresse> \
> netmask <Netzmaske>
```

geschehen.

Ist z. B. der Rechner 192.168.0.42 über ein zweites Netzwerk-Interface mit dem Netz 192.168.3.x verbunden, so können wir ihn mit dem folgenden Aufruf als Gateway eintragen:

```
# route add -net 192.168.3.0 gw 192.168.0.42 \
> netmask 255.255.255.0
```

Sind Sie z. B. über eine Standleitung mit dem Internet verbunden, so hat Ihnen Ihr Provider üblicherweise einen Router zur Verfügung gestellt, der die Anbindung an den Provider realisiert und von Ihnen über ein normales Ethernetkabel angesprochen werden kann. Dieser stellt damit das Gateway für alle Pakete in das Internet dar.

Hier können wir uns das Leben noch einfacher machen. Es genügt ein Eintrag, der besagt: »Schicke alle Pakete, auf die kein anderer Eintrag paßt, an den Router.« Dies geschieht in der folgenden Weise:

```
# route add default gw <Gatewayadresse>
```

Eine Angabe von Zieladresse und Netzmaske entfällt und wird durch den Parameter »default« ersetzt. Hat unser Router also z. B. die Adresse 192.168.0.42, so lautet der Aufruf:

```
# route add default gw 192.168.0.42
```

Es ist auch möglich, Einträge aus der Routing-Tabelle zu löschen. Dazu ersetzt man einfach den Parameter »add« durch »del«.

Um nun die Konfiguration zu automatisieren, sollten Sie ein Runlevel-Skript erstellen, das die nötigen Aufrufe beim Booten tätigt. Nennen Sie es aber bitte nicht »route« oder »network«. Skripte dieses Namens existieren wahrscheinlich schon. Ich habe hier aus diesem Grund »ethdevs« genommen:

```
#!/bin/sh
#####################################################################
#
# /etc/init.d/ethdevs
#
#  Usage: /etc/init.d/ethdevs {start|stop}
#
#      Runlevel-Skript, um die Netzwerkkarten zu konfigurieren
#
# Changelog
#
# 2005-06-24 Jetzt können beliebig viele Karten konfiguriert
#            werden.
#
#            Eine eigene Variable erlaubt es, einzustellen,
#            ob lo konfiguriert werden soll.
#
# 2005-06-06 Wenn /etc/rc.config nicht existiert, wird statt dessen
#            /etc/rc.status benutzt
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
#####################################################################

### BEGIN INIT INFO
# Provides:          ethdevs
# Required-Start:    $local_fs procconf
# Should-Start:      pf-ipt pf-ipc dmz-ipt dmz-ipc
# Required-Stop:
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Richtet Netzwerkkarten ein
# Description:       Richtet Netzwerkkarten ein
### END INIT INFO
```

```
# ============================================================
# Zuerst definieren wir noch Variablen für
# die Kartenparameter (Bitte anpassen!)
# ============================================================

# - Soll Loopback (127.0.0.1) konfiguriert werden?
#   Oft macht dies die Distribution selbst.
#   (yes/ja oder no/nein)

DO_LOOPBACK=ja

# - Default-Gateway (leer für keines)

ETH_GW="192.168.1.42"

# - Anzahl der Karten, die Sie konfigurieren wollen

ETH_NUM=2

# - Karte 1

ETH_DEV_1="eth0"
ETH_IP_1="192.168.0.1"
ETH_MASK_1="255.255.255.0"
ETH_BCAST_1="192.168.0.255"

# - Karte 2

ETH_DEV_2="eth2"
ETH_IP_2="192.168.1.1"
ETH_MASK_2="255.255.255.0"
ETH_BCAST_2="192.168.1.255"

# - Karte 3

ETH_DEV_3=""
ETH_IP_3=""
ETH_MASK_3=""
ETH_BCAST_3=""

# ============================================================
# Nun übernehmen wir all diese SuSE-Variablen, insbesondere
# rc_done und rc_failed
# ============================================================

if test -f /etc/rc.config
then
        . /etc/rc.config
elif test -f /etc/rc.status
then
        . /etc/rc.status
fi

if test "$rc_done" == ""
then
        rc_done="\t\t\t\t\t\t\t o.k."
fi
```

```
if test "$rc_failed" == ""
then
        rc_failed="\t\t\t\t\t\t\t GING SCHIEF !!!!"
fi

# ============================================================
# Nun geht's los
# ============================================================

# Sehen wir mal, wie wir aufgerufen wurden ...

case "$1" in
start)
    return="$rc_done"

    echo -n "Konfiguration von lo:"
    case "$DO_LOOPBACK" in
    ja)
        ifconfig lo 127.0.0.1 netmask 255.0.0.0 \
        up || return="$rc_failed"
        ;;
    yes)
        ifconfig lo 127.0.0.1 netmask 255.0.0.0 \
        up || return="$rc_failed"
        ;;
    esac
    echo -e "$return"

    num=1
    while test $num -le  "$ETH_NUM"
    do
        return="$rc_done"
        curip="ETH_IP_$num"
        curip="${!curip}"

        if test "$curip" != ""
        then
            return="$rc_done"

            curdev="ETH_DEV_$num"
            curdev="${!curdev}"

            curmask="ETH_MASK_$num"
            curmask="${!curmask}"

            curbcast="ETH_BCAST_$num"
            curbcast="${!curbcast}"

            echo -n "Konfiguration von $curdev:"

            ifconfig "$curdev" "$curip" netmask "$curmask" \
            broadcast "$curbcast" up || return="$rc_failed"

            echo -e "$return"
        fi
        num=$((num + 1))
    done
```

```
    if test "$ETH_GW" != ""
    then
        return="$rc_done"
        echo -n "Setzen der Default-Route:"
        route add default gw "$ETH_GW" || return="$rc_failed"
        echo -e "$return"
    fi

    ;;
stop)
    if test "$ETH_GW" != ""
    then
        return="$rc_done"
        echo -n "Löschen der Default-Route:"
        route del default gw "$ETH_GW" || return="$rc_failed"
        echo -e "$return"
    fi

    num=1
    while test $num -le  "$ETH_NUM"
    do
        curip="ETH_IP_$num"
        curip="${!curip}"
        if test "$curip" != ""
        then
            return="$rc_done"
            curdev="ETH_DEV_$num"
            curdev="${!curdev}"

            echo -n "Herunterfahren von $curdev:"
            ifconfig "$curdev" down || return="$rc_failed"
            echo -e "$return"
        fi
        num=$((num + 1))
    done

    echo -n "Herunterfahren von lo:"
    case "$DO_LOOPBACK" in
    ja)
        ifconfig lo 127.0.0.1 down || return="$rc_failed"
        ;;
    yes)
        ifconfig lo 127.0.0.1 down || return="$rc_failed"
        ;;
    esac
    echo -e "$return"

    ;;

*)
    echo "Usage: $0 {start|stop}"
    exit 1

   ;;
esac
```

Dieses Beispielskript kann man über Variablen konfigurieren, um es an seine persönlichen Bedürfnisse anzupassen. Die Variablen bedeuten im einzelnen:

DO_LOOPBACK Ist diese Variable auf »yes« oder »ja« gesetzt[1], so wird das Interface `lo` aktiviert und als Adresse 127.0.0.1 eingetragen. Normalerweise ist dies nicht notwendig, da die Distributionen dies automatisch tun. Haben Sie es aber geschafft, tatsächlich alle netzwerkbezogenen Runlevel-Skripte zu entfernen, dann brauchen Sie den Parameter vielleicht.

ETH_GW In dieser Variable steht das Default-Gateway. Dies ist nur dann relevant, wenn Sie über einen Router mit dem Internet verbunden sind. Benutzen Sie ein Modem, ISDN oder DSL, dann lassen Sie diese Variable bitte leer.

ETH_NUM Hiermit geben Sie an, wieviele Netzwerkkarten[2] Sie mit diesem Skript konfigurieren wollen.

ETH_DEV_n Falls n größer als eins und kleiner oder gleich *ETH_NUM* ist, dann sollte es einen Namen eines Netzwerk-Interfaces enthalten, das Sie konfigurieren wollen (z. B. `eth0`). Ist die Variable allerdings leer, so wird mit *ETH_DEV_(n+1)* fortgefahren.

ETH_IP_n Diese Variable enthält die IP-Adresse für *ETH_DEV_n*.

ETH_MASK_n Dies ist die Netzwerkmaske für *ETH_DEV_n*.

ETH_BCAST_n Hierin geben wir schließlich die Broadcast-Adresse für *ETH_DEV_n* an.

Unter SuSE müssen wir noch ein paar Klimmzüge machen. Dort existiert ein Runlevel-Skript `network`, das Sie nicht einfach löschen oder deaktivieren können, ohne die natürliche Ordnung der Dinge im SuSE-Universum durcheinanderzubringen. Sie können also `network` nicht einfach durch `ethdevs` ersetzen. Sie müssen aber sicherstellen, daß SuSE weiß, daß `procconf` und die Firewall-Skripte vor `network` aufgerufen werden sollen.

Sie haben nun zwei Möglichkeiten:

1. Sie konfigurieren die Netzwerkkarten hauptsächlich über `ethdevs`.

   In diesem Fall dürfen die Karten im `yast` nicht konfiguriert sein. Idealerweise haben Sie bereits bei der Installation darauf geachtet, daß dies nicht passiert. Andernfalls müssen Sie die Konfiguration der Karten im `yast` löschen.

   Tragen Sie nun in `/etc/init.d/network` in der Zeile `# Required-Start:` zusätzlich `ethdevs` ein:

   ```
   # Required-Start: $local_fs ethdevs
   ```

   Dadurch wird `ethdevs` vor `network` gestartet, womit sichergestellt ist, daß auch die anderen Skripte in der richtigen Reihenfolge gestartet werden.

   Die Variable *DO_LOOPBACK* setzen Sie bitte auf `nein` oder `no`.

1 Die Groß-/Kleinschreibung ist in diesem Fall wichtig.

2 Hier zählen nur die Netzwerkkarten, die Sie im folgenden definieren. Haben Sie *DO_LOOPBACK* gesetzt, so zählt `lo` nicht mit, sondern Sie geben nur die echten Ethernet-Karten an.

2. Sie konfigurieren die Karten über den yast.

   In diesem Fall benötigen Sie `ethdevs` nicht.

   Tragen Sie aber unbedingt im Skript `/etc/init.d/network` in der Zeile `# Required-Start:` zusätzlich `procconf` ein:

   ```
   # Required-Start: $local_fs procconf
   ```

   Auf diese Weise ist sichergestellt, daß zuerst das */proc*-Dateisystem und die Firewall-Regeln konfiguriert werden, bevor die Netzwerkkarten aktiviert werden und damit Kontakt zur Außenwelt besteht.

Aktivieren Sie das Skript aber bitte jetzt noch nicht, sondern konfigurieren Sie erst noch die restlichen Netzwerkeinstellungen, die in den folgenden Abschnitten beschrieben werden, und führen Sie dann die in Kapitel 10, Abschnitt *Verbindungstests*, ab Seite 257 beschriebenen Tests durch, um sicherzustellen, daß auch alles wie gewünscht funktioniert.

Erst dann sollten Sie das Skript verlinken bzw. `insserv` aufrufen.

Einrichten von Modem oder DSL

Sowohl der Zugang über Modem als auch über DSL wird mit Hilfe des PPP-Daemons (`pppd`) gesteuert. Lediglich die Konfiguration unterscheidet sich in einigen Details. Im folgenden wollen wir daher einmal die Grundkonfiguration des `pppd` betrachten, bevor wir dann auf die Unterschiede zwischen der Modem- und der DSL-Konfiguration eingehen.

Der PPP-Daemon

Für die Ansteuerung eines Modems ist der PPP-Daemon (`pppd`) zuständig. Dieser wird über mehrere Dateien konfiguriert. Dies sind im einzelnen:

/etc/ppp/options Generelle Optionen.

~/.ppprc Optionen, die gelten sollen, wenn ein bestimmter Benutzer den `pppd` aufruft. Bestimmte Optionen sind privilegiert. Diese dürfen von normalen Benutzern nicht ausgewählt werden und können daher auch nicht in dieser Datei stehen.

/etc/ppp/options.<ttyname> Die Optionen in dieser Datei sind spezifisch für das Device, an das das Modem angeschlossen ist.

/etc/peers/<name> Diese Datei erlaubt es, Optionen für einen bestimmten Verbindungspartner festzulegen. So kann man für jeden Internet-Provider eine eigene Datei mit Einstellungen festlegen. Angewählt wird diese dann, indem man den `pppd` mit dem Argument `call` *<name>* aufruft.

/etc/ppp/pap-secrets Namen und Paßwörter für die Authentisierung im Password Authentication Protocol. Diese stehen im Klartext in der Datei. Auch wird das Paßwort im Klartext an die Gegenstelle geschickt.

/ets/ppp/chap-secrets Namen und Paßwörter für die Authentisierung im Challenge Handshake Authentication Protocol. Diese stehen im Klartext in der Datei. Hierbei wird das Paßwort selbst nicht übermittelt, sondern nur benutzt, um eine Antwort auf eine zufällig gewählte Anfrage zu generieren (Challenge-Response-Verfahren). Dies ist das neuere der beiden Verfahren.

Außerdem existieren diverse Dateien, die – falls vorhanden und ausführbar – ausgeführt werden, wenn eine Verbindung auf- oder abgebaut wird. Der `pppd` wartet allerdings nicht, bis sie beendet wurden. Eine vollständige Liste finden Sie auf der Manpage des `pppd` (`man pppd`). Zwei Dateien verdienen aber besondere Aufmerksamkeit:

/etc/ppp/ip-up Dieses Skript wird gestartet, wenn die Verbindung aufgebaut ist und IP-Pakete gesendet werden können.

/etc/ppp/ip-down Dies ist das Gegenstück, es wird ausgeführt, wenn keine IP-Pakete mehr gesendet werden können.

Beide Skripte werden mit den Parametern

Interface TTY Baudrate Lokale_Adresse Gateway_Adresse

aufgerufen. Dies kann man z. B. dazu benutzen, in den Dateien Firewallregeln nachzutragen, die sich auf die IP-Adresse des externen Interfaces beziehen, falls Ihnen Ihre Adresse vom Provider dynamisch zugewiesen wird.

Wenn bereits Skripte oder Konfigurationsdateien in */etc/ppp* existieren, dann sollten wir sie umbenennen oder an einen anderen Ort kopieren, damit sie nicht den von uns neu erstellten in die Quere kommen. Dies gilt insbesondere für */etc/ppp/ip-up* und */etc/ppp/ip-down*.

Beginnen wir mit den Grundeinstellungen in */etc/ppp/options* :

```
#
# Modemgrundeinstellung

asyncmap 0

#
# Auf keinen Fall "ubertragene Pa"sw"orter
# protokollieren

hide-password

#
# Die eigene Adresse nicht raten,
# sie muß von der Gegenstelle vergeben werden

noipdefault

#
# Keine Verbindungsaufnahme ohne Authentisierung
# der Gegenstelle (mu"s f"ur Internet-Provider
# im entsprechenden Skript abgeschaltet werden)

auth
```

```
#
# Sende alle 30 sec eine LCP-Echo-Anfrage, kommt
# viermal keine Antwort, so hat die Gegenstelle
# wohl aufgelegt

lcp-echo-interval 30
lcp-echo-failure 4

#
# Die Firewall ist nur f"ur die Vermittlung von
# IP-Paketen zust"andig. IPX w"urde nur L"ocher
# in unsere Schutzma"snahmen rei"sen.

noipx
```

Hier legen wir in erster Linie grundsätzliche Parameter fest, die für jede Verbindung gelten sollen. So wollen wir nicht, daß im Systemprotokoll die übertragenen Paßwörter mitgeschrieben werden. Weiterhin fordert der Eintrag `auth`, daß sich jede Gegenstelle erst authentisieren muß. Da Internet-Provider dies in der Regel nicht tun, muß diese Einstellung in den konkreten Optionen für den jeweiligen Provider wieder abgestellt werden.

Modemkonfiguration

Definieren wir nun den Verbindungsaufbau zu einem bestimmten Provider. Wir gehen einmal von einem Provider »Dummynet« aus, der die Telefonnummer 12345678 hat. Da dieser aber nicht für jedermann geeignet ist, habe ich im Anhang A ab Seite 595 eine Liste mit Call-by-Call-Anbietern zusammengestellt, die keine Anmeldung verlangen und über die normale Telefonrechnung abrechnen.

Je nach dem verwendeten Provider müssen Sie dabei die folgenden Daten an den entsprechenden Stellen in den Beispielen anpassen:

- die verwendete serielle Schnittstelle (*ttyS0* (COM1), *ttyS1* (COM2), ...)
- die anzurufende Telefonnummer
- den zu verwendenden Anmeldenamen
- das Anmeldepaßwort

Bevor wir aber nun mit der eigentlichen Konfiguration beginnen, sollten wir einmal ausprobieren, ob wir das Modem richtig angeschlossen haben und es korrekt funktioniert.

Dazu müssen wir das Modem als erstes an Rechner und Telefondose anschließen. Solange wir den `pppd` noch nicht gestartet haben, ist dies kein Problem, da noch keine Möglichkeit besteht, eine IP-Verbindung herzustellen.

Wir wollen nun versuchen, unseren Provider manuell anzurufen und zu sehen, ob unter der Nummer tatsächlich ein Einwahlrechner antwortet. Dies geschieht am einfachsten mit einem Terminalprogramm wie `minicom`.

Ruft man dieses als root mit

```
# minicom -s
```

auf, so öffnet sich beim Start des Programms ein Menü, mit verschiedenen Punkten zur Konfiguration des Programms. Unter »Serial Port Setup« können wir nun mit <A> die verwendete serielle Schnittstelle (*/dev/ttyS0*[3], ...) einstellen. Mit <Return> gelangen wir wieder zurück in das Hauptmenü.

Mit »Save setup as dfl« erklären wir diese Einstellung zum Standard, so daß wir beim nächsten Start von minicom die Schnittstelle nicht mehr konfigurieren müssen und daher auf den Parameter -s verzichten können.

Wählen wir nun »Exit«, so wird das Modem initialisiert, und wir können ihm Befehle senden. Bevor wir dies allerdings tun, bietet es sich an, die Einstellung »Local echo« durch Drücken von *<Strg>A* und *E* auf »ON« zu setzen. Damit werden von uns eingegebene Zeichen nicht nur an das Modem gesendet, sondern auch vom Programm selbst am Bildschirm angezeigt.

Senden wir nun z. B. »at«, so sollte das Modem mit OK antworten. Bleibt der Bildschirm dagegen leer, so haben wir uns vermutlich in der zu verwendenden seriellen Schnittstelle geirrt. Mit *<Strg>A* und *O* gelangen wir zurück in das Konfigurationsmenü, wo wir den Fehler korrigieren können.

Ist die richtige Schnittstelle eingestellt, so können Sie das Modem nun anweisen zu wählen. Das geschieht mit »atdt« (Tonwahl) bzw. »atdp« (Pulswahl), gefolgt von der Telefonnummer. Nun sollte sich Ihnen nach langem Gepiepe in etwa folgendes Bild bieten:

```
atdp12345678
CONNECT 49333/LAPM
~ÿ}#.!}!.} }9}"}&} }*} }
NO CARRIER
```

Die Meldung CONNECT besagt, daß auf der anderen Seite ein Modem den Anruf entgegengenommen hat. Die seltsamen Zeichen, die darauf folgen, sind der Versuch der Gegenseite, eine PPP-Verbindung zu initiieren. Im obigen Beispiel habe ich darauf nicht reagiert. Darum hat die Gegenseite aufgegeben und aufgelegt. Das Modem zeigt dies mit der Meldung NO CARRIER an. Wenn dieser automatische Abbruch zu lange dauert und der Bildschirm sich mit unverständlichen Zeichen zu füllen beginnt, dann können Sie die Verbindung auch von sich aus mit *<Strg>A* und *H* abbrechen.

Einen Sonderfall stellen noch Server dar, die neben PPP-Verbindungen auch Terminal-Verbindungen über Modem entgegennehmen. Bei diesen würde die Einwahl etwa so aussehen:

```
atdp12345678
CONNECT 49333/LAPM
** Welcome to Miscatonic University ***
Login:
```

3 Im minicom müssen Sie den kompletten Pfad für das Device angeben. Bei der Konfiguration des pppd reicht die Angabe des Dateinamens ohne */dev/*.

Hier sollte es möglich sein, durch Eingabe von Benutzername und Paßwort ebenfalls so weit zu kommen, daß wieder die kryptischen Zeichen angezeigt werden, die auf den PPP-Verbindungsaufbau hinweisen. Ist dies nicht der Fall wenden Sie sich bitte an den Kundendienst des jeweiligen Providers.

Nun können Sie `minicom` mit *<Strg>A* und *X* verlassen. Bitte ziehen Sie nun wieder den Stecker aus der Telefondose.

Nachdem Sie wissen, daß Sie Ihren Provider problemlos erreichen können, gilt es nun, eine providerspezifische Konfigurationsdatei zu erstellen. Wir wollen dabei davon ausgehen, daß der `pppd` nur einmal vom System gestartet wird, um dann bei Bedarf automatisch zu wählen.

Hier nun die providerspezifischen Optionen in */etc/ppp/peers/dummynet.modem*:

```
# Der Modemanschlu"s

ttyS0 115200
crtscts
lock
modem

#
# Dummy-Adressen, diese sollten im lokalen Netz noch nicht vergeben sein!

10.1.0.1:10.1.0.2

#
# Dieses Kommando regelt die Kommunikation mit dem Modem

connect '/usr/sbin/chat -v -f /etc/ppp/peers/dummynet.chat'

#
# Provider wollen sich in der Regel uns gegen"uber nicht authentisieren.

noauth

#
# Zur Fehlersuche, sp"ater auskommentieren:

debug
nodetach

#
# F"ur den automatischen Verbindungsaufbau

ipcp-accept-local
ipcp-accept-remote
demand
defaultroute

#
# Wenn 10 min keine IP-Pakete empfangen werden,
# kann die Verbindung abgebaut werden.

idle 600
```

```
#
# Wie viele Sekunden soll das Modem warten, wenn
# die Leitung besetzt ist?

holdoff 40

#
# Anmeldename f"ur CHAP und PAP

name "nobody"
```

Zu Beginn geben wir das Device (hier: `ttyS0`, also COM1) sowie ein paar Standardoptionen für Modems an. Dann werden zwei IP-Adressen festgelegt. Sie werden für das vom `pppd` bereitgestellte Interface und für die Adresse der Gegenstelle beim Provider verwendet, solange noch keine Verbindung zustande gekommen ist. Wichtig hierbei ist, daß Sie Adressen verwenden, die noch nicht in Ihrem lokalen Netz verwendet werden.

Die Option `debug` bewirkt eine ausführliche Protokollierung der Kommunikation im Systemprotokoll. Steigern ließe sich dies nur noch durch `kdebug` *n*, das die Protokollierung im PPP-Treiber selbst aktiviert. Dann würden für

n*=*1 ausführlichere Meldungen,

n*=*2 alle empfangenen Pakete,

n*=*4 alle gesendeten Pakete

protokolliert. Dabei können die Optionen durch Addition kombiniert werden. `kdebug 7` wäre die höchstmögliche Protokollierungsstufe. Jedes Paket würde einen Eintrag im Systemprotokoll bewirken. Im laufenden Betrieb sollte man auf `kdebug` verzichten, da sonst das Systemprotokoll sehr schnell sehr groß wird.

Die Option `nodetach` bewirkt, daß der `pppd` nicht im Hintergrund gestartet wird, sondern weiterhin das virtuelle Terminal benutzt, auf dem er gestartet wurde. Das bedeutet, daß man, während er aktiv ist, ein anderes Terminal benutzen muß (<Alt><Fn>). Allerdings erfolgt auf diese Weise eine Protokollierung direkt auf den Bildschirm, und der Daemon läßt sich einfach durch Drücken von <Ctrl><c> beenden. In der ersten Testphase ist diese Einstellung daher sehr praktisch, später wird man sie aber auskommentieren.

Die Option `holdoff` definiert, wie viele Sekunden bei besetzter Leitung gewartet werden soll, bevor erneut gewählt wird. Hierbei ist zu beachten, daß Modems üblicherweise eine Wahlsperre eingebaut haben, die es je nach Hersteller verbietet, z. B. häufiger als alle 30 Sekunden zu wählen. Die Angabe sollte passend gewählt werden.

Die Option `name` legt fest, unter welcher Kennung sich der Rechner beim Provider anmeldet. In diesem Fall nennen wir uns `nobody`, da Dummynet sich wie einige andere Call-by-Call-Provider nicht wirklich für unsere Kennung interessiert. Bei anderen Providern, insbesondere solchen, die eine Anmeldung verlangen, wird dem Benutzer dagegen explizit eine Kennung zugeteilt. Bei T-Online ist dies z. B. eine ziemlich lange Kette von Zahlen und Sonderzeichen, die folgendermaßen aufgebaut ist:

<*Anschlußkennung*><*T-Online-Nr.*>#<*Mitbenutzernummer*>`@t-online.de`

Da diese Kennung das Zeichen # enthält, muß der Name unbedingt, wie oben dargestellt, in Anführungszeichen eingeschlossen werden. Andernfalls würde der pppd das # als Kommentarzeichen ansehen und es zusammen mit den nachfolgenden Zeichen ignorieren.

Die Anwahl des Providers erledigt der pppd nicht selbst, sondern überläßt sie einem externen Programm. Dieses wird mit der Option »connect« bestimmt. In unserem Fall haben wir uns für chat entschieden, das zum Umfang des pppd gehört. Wir rufen es mit den Optionen -v und -f *Datei* auf. Die erste sorgt für eine ausführlichere Protokollierung, letztere gibt an, welche Datei die Konfiguration für chat enthält. In unserem Fall ist dies die Datei */etc/ppp/peers/dummynet.chat*:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
""    "\ratz"
OK    "atdt12345678"
TIMEOUT 60
CONNECT ""
```

Hier werden chat einige einfache Regeln vorgegeben. Zuerst wird definiert, welche Rückmeldungen des Modems ein Mißlingen des Vorgangs bedeuten (ABORT). Dann folgen Regeln der Art

Rückmeldung Befehl

die der Reihe nach abgearbeitet werden. Jedesmal wartet chat auf eine Rückmeldung des Modems, worauf es dann einen neuen Befehl schickt. Um dies in Gang zu bringen, wird als erste Rückmeldung eine leere Zeichenkette erwartet, worauf der Befehl atz sofort gesendet wird. Das »\r« steht hierbei für die Zeilenendetaste (CR). Diese Folge setzt das Modem in seinen Grundzustand zurück. Es sollte nun bereit für die nachfolgenden Befehle sein.

Ist dies der Fall, so wird es mit OK antworten, worauf wir es mit dem Befehl atdt die Nummer unseres Providers wählen lassen. Da es jetzt eine Weile dauern kann, bis sich unser Modem und das des Providers auf eine Übertragungsgeschwindigkeit geeinigt haben, geben wir chat 60 Sekunden Zeit (TIMEOUT 60), auf eine Erfolgsmeldung (CONNECT) zu warten. Erhalten wir diese, ist die Anwahl beendet, wir geben also keinen weiteren Befehl (""). Allerdings wird auch für eine leere Zeichenkette immer ein Zeilenende gesendet.

Zu einer Anmeldung gehört neben einer Benutzerkennung auch ein Paßwort. Die meisten Provider benutzen CHAP oder PAP, um es zu übermitteln. Allerdings kommt es gerade im Universitätsbereich vor, daß nach dem Verbindungsaufbau erst einmal nach einer Benutzerkennung und einem Paßwort gefragt wird, bevor das PPP gestartet wird. Verbindet man sich mit so einem Rechner mit einem Terminalprogramm, so könnte eine Einwahl folgendermaßen aussehen:

```
*** Welcome to Miscatonic University ***
Login: lessig
Password: *****
~y}#.!}!}!} }8}!}$}%U}"}&} } } } }%}& ...}'}"}(}"} .~~y}
```

Die kryptischen Zeichen zeigen den Aufbau der PPP-Verbindung an. Hier ist es nötig, weitere Zeilen an die Datei */etc/ppp/peers/dummynet.chat* anzufügen:

```
ogin:--ogin: "lessig"
assword: "SeCRet"
```

Abgesehen davon, daß in den Mustern das erste Zeichen ausgelassen ist[4], fällt die besondere Form `ogin:--ogin:` auf. Dies bewirkt, daß im Falle des Ausbleibens von ... ogin: ein leerer String (Zeilenende)[5] gesendet und noch einmal auf `...ogin:` gewartet wird.

Bei PAP und CHAP wird das Paßwort in einer eigenen Konfigurationsdatei festgelegt, die auch noch spezifisch für das verwendete Anmeldeverfahren ist. Für das veraltete PAP ist dies die Datei */etc/ppp/pap-secrets*, die z. B. so aussehen könnte:

```
"nobody"    *    "SeCRet"
```

Die entsprechende Datei für das mittlerweile gebräuchliche CHAP heißt dagegen */etc/ppp/chap-secrets*, ist aber gleich aufgebaut:

```
"nobody"    *    "SeCRet"
```

In der ersten Spalte steht dabei die Kennung des Benutzers, in der zweiten der Name des Rechners, an dem die Anmeldung erfolgen soll, in der letzten schließlich das Paßwort im Klartext. Da die zu benutzende Kennung schon in der Datei */etc/ppp/peers/dummynet.modem* vorgegeben wird, können in der Datei durchaus mehrere Zeilen stehen. Den Namen des Providerrechners wird man in der Regel nicht kennen. Aus diesem Grund ist hier mit »*« vorgegeben, daß jeder Name akzeptiert wird.

Auch hier sind der Name und das Paßwort wieder in Anführungszeichen eingeschlossen. Dies ist insbesondere dann notwendig, wenn die jeweilige Zeichenkette wie im Fall einer T-Online-Kennung Sonderzeichen enthält. Es schadet aber nicht, die Anführungszeichen grundsätzlich zu verwenden.

Bis jetzt haben wir den `pppd` zwar konfiguriert, er wird aber noch nicht gestartet. Im folgenden werden Sie sehen, wie das geschehen kann. Stecken Sie aber jetzt nicht einfach den Stecker in die Telefondose, um zu versuchen, ob es funktioniert. Konfigurieren Sie bitte erst die Namensauflösung im Netzwerk, wie dies im folgenden beschrieben ist, und benutzen Sie dann das in Kapitel 10, Abschnitt *Verbindungstests*, ab Seite 257 beschriebene Verfahren, um die Modemanbindung zu testen.

Dort wird ein Skript beschrieben, das erst einmal eine sichere Umgebung aufbaut und Sie dann auffordert, den `pppd` zu starten. Wechseln Sie dazu in eine andere Konsole, und starten Sie den Daemon manuell mit einem Aufruf der Art

4 Das erste Zeichen der Gegenstelle kann schon einmal verlorengehen.

5 Die zu sendende Zeichenkette steht zwischen den beiden »-«-Zeichen.

```
# /usr/sbin/pppd call dummynet.modem
```

Dabei sollten Sie `dummynet.modem` gegebenenfalls durch den Namen der Optionendatei in */etc/ppp/peers* ersetzen, die Sie benutzen wollen.

Wechseln Sie nun in die Konsole zurück, in der Sie das Skript gestartet haben, und bestätigen Sie, daß Sie fertig sind. Wenn Sie aufgefordert werden, die Verbindung zu beenden, wechseln Sie bitte wieder in die Konsole, in der Sie den `pppd` gestartet haben. Da momentan noch der Parameter `nodetach` in der Konfigurationsdatei steht, hat der `pppd` seine Verbindung zum virtuellen Terminal nicht aufgegeben und kann daher durch Eingabe von <Ctrl><c> beendet werden.

Das Skript wird sich jetzt auch beenden. Sie können diesen Vorgang ruhig mehrere Male mit verschiedenen Zielrechnern wiederholen, solange Sie das Skript stets starten, bevor Sie die Netzwerkverbindung herstellen.

Haben wir den Verbindungsaufbau getestet, so können wir die Parameter `nodetach` und `debug` aus */etc/ppp/peer/dummynet.modem* entfernen und den Aufruf in einem Runlevel-Skript unterbringen.

Dies könnte z. B. folgendermaßen aussehen:

```
#!/bin/sh
###############################################################################
#
# /etc/init.d/modem
#
#  Aufruf: /etc/init.d/modem {start|stop}
#
#      Runlevel-Skript, um den pppd zu starten
#
# Changelog
#
# 2005-06-06 Wenn /etc/rc.config nicht existiert, wird statt dessen
#            /etc/rc.status benutzt
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder h"ohere Version
#
###############################################################################

### BEGIN INIT INFO
# Provides:          modem
# Required-Start:    $local_fs procconf
# Should-Start:      pf-ipt pf-ipc dmz-ipt dmz-ipc
# Required-Stop:
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Richtet ein Modem ein
# Description:       Richtet ein Modem ein
### END INIT INFO
```

```
# Eine kleine Routine, um zu überprüfen, ob ein Dienst gerade läuft

checkserv()
{
        ps ax | grep "$1" | grep -v grep > /dev/null
}

# Zuerst übernehmen wir all diese SuSE-Variablen, insbesondere
# rc_done und rc_failed

if test -f /etc/rc.config
then
        . /etc/rc.config
elif test -f /etc/rc.status
then
        . /etc/rc.status
fi

if test "$rc_done" == ""
then
        rc_done="\t\t\t\t\t\t\t o.k."
fi

if test "$rc_failed" == ""
then
        rc_failed="\t\t\t\t\t\t\t GING SCHIEF !!!!"
fi

# Eine Variable für den Erfolg

return=$rc_done


# Nun sehen wir mal, wie wir aufgerufen wurden

case "$1" in
start)
        echo -n "Starting pppd:"
        /usr/sbin/pppd call dummynet.modem 2>/dev/null
        checkserv pppd || return=$rc_failed
        echo -e "$return"
        ;;
stop)
        echo -n "Stopping pppd:"
        killall -TERM /usr/sbin/pppd 2>/dev/null
        checkserv pppd && return=$rc_failed
        echo -e "$return"
        ;;
*)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
```

Nun können Sie das Skript verlinken oder `insserv` aufrufen. Vorher sollten Sie es aber erst einmal von Hand mit dem Parameter `start` bzw. `stop` aufrufen, um sicherzustellen, daß sich keine Fehler in das Skript eingeschlichen haben.

DSL

Wenn Sie regelmäßig große Datenmengen aus dem Internet herunterladen wollen, so bietet sich DSL als Zugangsmethode an. Im Gegensatz zu anderen Verfahren wählen Sie sich bei DSL nicht beim Provider ein und blockieren dabei für die Dauer Ihres Aufenthaltes im Internet eine Leitung, sondern Sie befinden sich gewissermaßen im Datennetz des Providers und senden und empfangen einzelne Datenpakete bei Bedarf.

Technisch ist dies so realisiert, daß Ihre Firewall über ein normales Ethernetkabel mit einem DSL-Modem verbunden ist, das die eingehenden Pakete so umsetzt, daß sie über die Telefonleitung übertragen werden können. Die Datenübertragung zwischen Ihrer Firewall und dem DSL-Modem ist allerdings etwas komplizierter als zwischen zwei normalen Rechnern im lokalen Netz.

Es kommt ein Protokoll namens *PPP over Ethernet* (PPPoE) zum Einsatz. Dabei werden PPP-Pakete, wie sie sonst bei der Einwahl mit dem Modem verwendet werden, in Ethernet-Pakete verpackt, anstatt sie über eine serielle Leitung zu senden. Dies hat für den Provider den Vorteil, daß Sie wie bei einer Einwahl über Modem eine virtuelle Verbindung aufbauen. Dabei müssen Sie sich an seinem Einwahlrechner erst mit Name und Paßwort anmelden, bevor Sie Kontakt mit dem Internet aufnehmen können. Dadurch ist klar definiert, wann Sie online sind und wann nicht. Dies erlaubt es dem Provider, zeitbasiert abzurechnen.

Es existieren zwei Varianten eines DSL-Anschlusses. Manche Provider stellen Ihnen nicht nur ein DSL-Modem, sondern auch einen Router zur Verfügung. Eine solche Anbindung entspricht aus Sicht Ihrer Firewall einer direkten Netzwerkanbindung über Ethernet, wie sie in Kapitel 10, Abschnitt *Konfiguration der Netzwerkkarte*, ab Seite 219 beschrieben ist.

Wir wollen uns hier daher auf die direkte Anbindung eines DSL-Modems konzentrieren. Hier müssen Sie Ihren Rechner dazu bringen, mit PPPoE zu kommunizieren.

Alles, was wir dazu brauchen, ist ein Dienst, der aus Sicht des `pppd` eine serielle Schnittstelle darstellt und der die PPP-Pakete in Ethernet-Pakete »einpackt« und über eine Netzwerkkarte verschickt. Wir werden hier das Programm `pppoe` von Roaring Penguin Software Inc.[6] verwenden, das in den betrachteten Distributionen enthalten ist.

Außerdem erlaubt uns das Programm auch, zu testen, ob unser Telefonanschluß prinzipiell für DSL freigeschaltet ist und der Zugangsrechner unseres Providers erreichbar ist.

Dazu müssen wir jetzt erst einmal das DSL-Modem mit Rechner und Telefonnetz verbinden. Da der `pppd` noch nicht gestartet ist, kann noch keine IP-Verbindung zum oder aus dem Internet geöffnet werden.

Als erstes aktivieren wir das Netzwerk-Interface, an das unser DSL-Modem angeschlossen ist (hier: *eth0*):

```
# ifconfig eth0 up
```

6 *http://www.roaringpenguin.com*

Wenn wir nun nachsehen, werden wir feststellen, daß unser Netzwerk-Interface wie gewünscht aktiviert (UP) ist, aber keine IP-Adresse besitzt:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:80:AD:18:9C:97
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:9 Base address:0x300
```

Dies ist genau, was wir wollen. Für DSL ist an dieser Stelle keine IP-Adresse nötig. Das Ethernet-Device ist nur ein Hilfskonstrukt, das vom pppoe benötigt wird, der kein IP, sondern nur Ethernet spricht. Wenn wir den pppd starten, wird er ein eigenes Device (z. B. *ppp0*) erzeugen, das dann auch eine reguläre IP-Adresse besitzt, die dem Rechner vom Provider zugewiesen wurde.

Als nächstes bitten wir pppoe zu überprüfen, ob er einen Access-Concentrator erreichen kann. Dabei handelt es sich um einen Verbindungsrechner, der unsere PPPoE-Pakete entgegennimmt und ins Internet weitervermittelt. Dieses geschieht mit dem folgenden Befehl:

```
# pppoe -I eth0 -A
Access-Concentrator: KLNX12-erx
Got a cookie: 97 93 40 67 f0 b8 9c 99 0f ef 96 31 d0 32 94 2c
------------------------------------------------
AC-Ethernet-Address: 00:90:1a:10:0f:f3
------------------------------------------------
```

Die Option -I gibt dabei das zu verwendende Interface vor.

In diesem Beispiel wird ein Access-Concentrator namens KLNX12-erx gefunden. Wird Ihr pppoe an dieser Stelle nicht fündig, so haben Sie ein gravierendes Problem. Entweder sind die Kabel nicht richtig angeschlossen, oder Ihr Anschluß ist nicht für DSL freigeschaltet.

Nun können wir versuchen, eine Verbindung zum Access-Concentrator aufzunehmen. Dazu benutzen wir den folgenden Befehl:

```
# pppoe -T 80 -I eth0 -D pppoe.log
~ÿ}#À!}!} }2}!}$}%Ô}#}$À#}%}&}?Üš+N?~
```

Die nun erscheinenden wirren Zeichen sind die eigentlichen PPP-Daten, die einen Verbindungsaufbau einleiten. Sie werden normalerweise vom pppd interpretiert.

Mit der Option -D haben wir pppoe angewiesen, die übertragenen PPPoE-Pakete in der Datei *pppoe.log* zu protokollieren. Wenn wir in der Datei nachsehen, sollte das etwa so aussehen:

```
# less pppoe.log
rp-pppoe-3.3
18:51:25.579 SENT PPPoE Discovery (8863) PADI sess-id 0 length 4
SourceAddr 00:80:ad:18:9c:97 DestAddr ff:ff:ff:ff:ff:ff
01 01 00 00                                      ....

18:51:25.743 RCVD PPPoE Discovery (8863) PADO sess-id 0 length 38
SourceAddr 00:90:1a:10:0f:f3 DestAddr 00:80:ad:18:9c:97
01 02 00 0a 4b 4c 4e 58 31 32 2d 65 72 78 01 01   ....KLNX12-erx..
00 00 01 04 00 10 97 93 40 67 f0 b8 9c 99 0f ef   ........@g......
96 31 d0 32 94 2c                                 .1.2.,

18:51:25.745 SENT PPPoE Discovery (8863) PADR sess-id 0 length 24
SourceAddr 00:80:ad:18:9c:97 DestAddr 00:90:1a:10:0f:f3
01 01 00 00 01 04 00 10 97 93 40 67 f0 b8 9c 99   ..........@g....
0f ef 96 31 d0 32 94 2c                           ...1.2.,

18:51:26.014 RCVD PPPoE Discovery (8863) PADS sess-id 3091 length 4
SourceAddr 00:90:1a:10:0f:f3 DestAddr 00:80:ad:18:9c:97
01 01 00 00                                      ....

18:51:27.024 RCVD PPPoE Session (8864) SESS sess-id 3091 length 20
SourceAddr 00:90:1a:10:0f:f3 DestAddr 00:80:ad:18:9c:97
c0 21 01 8b 00 12 01 04 05 d4 03 04 c0 23 05 06   .!...........#..
1f dc a8 2b                                       ...+
```

Das letzte Paket wird dabei noch einige Male wiederholt.

PADI steht dabei für *PPPoE Active Discovery Initiation*. Mit diesem Paket fordert pppoe vorhandene Access-Concentrators auf, sich zu melden. Das Paket ist dabei an die Ethernet-Broadcast-Adresse *ff:ff:ff:ff:ff:ff* adressiert.

Ein Access-Concentrator, der so ein Paket empfängt, wird dies mit einem PADO (*PPPoE Active Discovery Offer*) beantworten. In diesem teilt er dem anfragenden Rechner seinen Namen und seine MAC-Adresse mit.

Damit kann der anfragende Rechner sich nun einen der antwortenden Access-Concentrators aussuchen und ihn mit einem PADR (*PPPoE Active Discovery Request*) um eine Verbindung bitten. Normalerweise sollte dieser nun mit einem PADC (*PPPoE Active Discovery Confirmation*) die Öffnung der Verbindung bestätigen. Dabei wird der Verbindung auch eine eindeutige Nummer zugewiesen (hier: 3091).

Nun besteht eine PPPoE-Verbindung, über die die eigentlichen PPP-Pakete übertragen werden können. Dies geschieht in Form von Session-Paketen (SESS). Diese PPP-Pakete haben wir eben in Form der wirren Zeichen auf dem Bildschirm gesehen.

Wenn der Ablauf bei Ihnen anders war und keine PPP-Pakete gesendet wurden bzw. Fehlermeldungen auftraten, so liegt das Problem wahrscheinlich bei Ihrem Provider. Sie sollten dessen Support kontaktieren.

Nachdem wir uns vergewissert haben, daß wir prinzipiell in der Lage sind, Verbindung zu unserem Provider aufzunehmen, sollten wir nun das Netzwerk-Interface wieder deaktivieren:

```
# ifconfig eth0 down
```

Anschließend ziehen wir das Kabel wieder ab, das unsere Netzwerkkarte mit dem DSL-Modem verbindet. Wir werden die Verbindung wieder herstellen, wenn wir zum eigentlichen Einwahltest kommen.

Definieren wir nun den Verbindungsaufbau zu einem bestimmten Provider »Dummynet«.Die providerspezifischen Optionen legen wir in */etc/ppp/peers/dummynet.dsl* ab:

```
#
# Dummy-Adressen, diese sollten im lokalen Netz noch nicht vergeben
# sein!

10.1.0.1:10.1.0.2

#
# Dummy-Adressen nach Einwahl durch vom Provider vergebene ersetzen

ipcp-accept-local
ipcp-accept-remote

#
# Das Programm wird vom pppd gestartet, und alle Ein- und Ausgaben
# werden auf ein Pseudoterminal umgeleitet. Dieses Pseudoterminal
# benutzt der pppd dann statt einer seriellen Leitung.

pty "/usr/sbin/pppoe -I eth0 -T 80 -m 1452"

#
# Dieses Kommando regelt die Kommunikation mit dem Modem.

connect /bin/true

#
# Provider wollen sich in der Regel uns gegen"uber nicht
# authentisieren.

noauth

#
# Zur Fehlersuche, sp"ater auskommentieren:

debug
nodetach

#
# Die folgenden Zeilen dienen dem automatischen
# Verbindungsaufbau. Sie k"onnen auskommentiert
# werden, wenn eine permanente Verbindung
# gew"unscht wird (z.B. Flatrate)

demand
idle 600

#
# einige Standardeinstellungen

mtu 1492
mru 1492
```

```
#
# Wenn die Verbindung abbricht, automatisch
# eine erneute Einwahl versuchen

persist

#
# Automatisch eine Default-Route eintragen

defaultroute

#
# Anmeldename f"ur CHAP und PAP

name "nobody"
```

Zu Beginn werden zwei IP-Adressen festgelegt. Sie werden für das vom `pppd` bereitgestellte Interface und für die Adresse der Gegenstelle beim Provider verwendet, solange noch keine Verbindung zustande gekommen ist. Wichtig hierbei ist, daß Sie Adressen verwenden, die noch nicht in Ihrem lokalen Netz verwendet werden.

Mit `pty` legen wir fest, daß der `pppd` keine serielle Schnittstelle ansprechen soll. Er startet vielmehr ein externes Programm und leitet dessen Ein- und Ausgaben auf ein Pseudoterminal um. Dieses Pseudoterminal benutzt er dann zur Kommunikation. Wir haben hier den `pppoe` eingetragen. Dabei wird mit der Option `-I` festgelegt, welches Netzwerk-Interface für die DSL-Anbindung genutzt werden soll.

`connect` gibt einen Befehl an, der die eigentliche Einwahl beim Provider auslöst. Bei Verwendung eines Modems wäre dies ein Aufruf von `chat`, um dem Modem die Kommandos zu geben, um die Telefonnummer des Providers zu wählen. Für DSL ist dies nicht nötig, weshalb wir hier einen Befehl eingetragen haben, der nichts tut.

Die Option `debug` bewirkt eine ausführliche Protokollierung der Kommunikation im Systemprotokoll. Steigern ließe sich dies nur noch durch `kdebug` *n*, das die Protokollierung im PPP-Treiber selbst aktiviert. Dann würden für

n*=*1 ausführlichere Meldungen,

n*=*2 alle empfangenen Pakete,

n*=*4 alle gesendeten Pakete

protokolliert. Dabei können die Optionen durch Addition kombiniert werden. `kdebug 7` wäre die höchstmögliche Protokollierungsstufe. Jedes Paket würde einen Eintrag im Systemprotokoll bewirken. Im laufenden Betrieb sollte man auf `kdebug` verzichten, da sonst das Systemprotokoll sehr schnell sehr groß wird.

Die Option `nodetach` bewirkt, daß der `pppd` nicht im Hintergrund gestartet wird, sondern weiterhin das virtuelle Terminal benutzt, auf dem er gestartet wurde. Das bedeutet, daß man, während er aktiv ist, ein anderes Terminal benutzen muß (<Alt><Fn>). Allerdings erfolgt auf diese Weise eine Protokollierung direkt auf den Bildschirm, und der Daemon läßt sich einfach durch Drücken von <Ctrl><c> beenden. In der ersten Testphase ist diese Einstellung daher sehr praktisch, später wird man sie aber auskommentieren.

Die Optionen `demand` und `idle 600` benötigen Sie, wenn die Anwahl beim Provider nur dann erfolgen soll, wenn auch tatsächlich Daten zu übertragen sind. Sie bewirken, daß eine Einwahl erst erfolgt, wenn Daten vorliegen, und daß die Verbindung beendet wird, wenn 600 Sekunden (10 Min.) keine Daten mehr übertragen wurden. Haben Sie eine Flatrate und wollen Sie, daß der Rechner die Verbindung permanent offenhält, so kommentieren Sie diese Zeilen aus.

Die Option `name` legt fest, unter welcher Kennung sich der Rechner beim Provider anmeldet. Bei T-Online ist dies z. B. eine ziemlich lange Kette von Zahlen und Sonderzeichen, die folgendermaßen aufgebaut ist:

<Anschlußkennung><T-Online-Nr.>#<Mitbenutzernummer>`@t-online.de`

Da diese Kennung das Zeichen # enthält, muß der Name unbedingt wie oben dargestellt in Anführungszeichen eingeschlossen werden. Andernfalls würde der `pppd` das # als Kommentarzeichen ansehen und es zusammen mit den nachfolgenden Zeichen ignorieren.

Zu einer Anmeldung gehört neben einer Benutzerkennung auch ein Paßwort. Um es zu übermitteln, wird CHAP oder PAP verwendet. Bei beiden wird das Paßwort in einer eigenen Konfigurationsdatei festgelegt, die auch noch spezifisch für das verwendete Anmeldeverfahren ist. Für das veraltete PAP ist dies die Datei */etc/ppp/pap-secrets*, die z. B. so aussehen könnte:

```
"nobody"    *    "SeCRet"
```

Die entsprechende Datei für das mittlerweile gebräuchliche CHAP heißt dagegen */etc/ppp/chap-secrets*, ist aber gleich aufgebaut:

```
"nobody"    *    "SeCRet"
```

In der ersten Spalte steht dabei die Kennung des Benutzers, in der zweiten der Name des Rechners, an dem die Anmeldung erfolgen soll, in der letzten schließlich das Paßwort im Klartext. Da die zu benutzende Kennung schon in der Datei */etc/ppp/peers/dummynet.dsl* vorgegeben wird, können in der Datei durchaus mehrere Zeilen stehen. Den Namen des Provider-Rechners wird man in der Regel nicht kennen. Aus diesem Grund ist hier mit »*« vorgegeben, daß jeder Name akzeptiert wird.

Auch hier sind der Name und das Paßwort wieder in Anführungszeichen eingeschlossen. Dies ist insbesondere dann notwendig, wenn die jeweilige Zeichenkette wie im Fall einer T-Online-Kennung Sonderzeichen enthält. Es schadet aber nicht, die Anführungszeichen grundsätzlich zu verwenden.

Bis jetzt haben wir den `pppd` zwar konfiguriert, er wird aber noch nicht gestartet. Im folgenden werden Sie sehen, wie das geschehen kann. Bitte stecken Sie aber jetzt nicht einfach den Stecker in das DSL-Modem und versuchen, ob es funktioniert. Konfigurieren Sie erst die Namensauflösung im Netzwerk, wie dies in den nächsten beiden Abschnitten beschrieben ist, und benutzen Sie dann das in Kapitel 10, Abschnitt *Verbindungstests*, ab Seite 257 dargestellte Verfahren, um die Anbindung zu testen.

Dort wird ein Skript beschrieben, das erst einmal eine sichere Umgebung aufbaut und Sie dann auffordert, den pppd zu starten. Wechseln Sie dazu in eine andere Konsole und starten den Daemon manuell mit einem Aufruf der Art

```
# /sbin/ifconfig eth0 up
# /usr/sbin/pppd call dummynet.dsl
```

Dabei sollten Sie dummynet.dsl gegebenenfalls durch den Namen der Optionsdatei in */etc/ppp/peers* ersetzen, die Sie benutzen wollen. Mit dem Befehl ifconfig aktivieren Sie das Netzwerk-Interface. Dabei ist es nicht nötig, ihm eine IP-Adresse zuzuweisen.

Wechseln Sie nun in die Konsole zurück, in der Sie das Skript gestartet haben, und bestätigen Sie, daß Sie fertig sind. Wenn Sie aufgefordert werden, die Verbindung zu beenden, wechseln Sie bitte wieder in die Konsole, in der Sie den pppd gestartet haben. Da momentan noch der Parameter nodetach in der Konfigurationsdatei steht, hat der pppd seine Verbindung zum virtuellen Terminal nicht aufgegeben und kann daher durch Eingabe von <Ctrl><c> beendet werden.

Das Skript wird sich jetzt auch beenden. Sie können diesen Vorgang ruhig mehrere Male mit verschiedenen Zielrechnern wiederholen, solange Sie das Skript stets starten, bevor Sie die Netzwerkverbindung herstellen.

Haben wir den Verbindungsaufbau getestet, so können wir die Parameter nodetach und debug aus */etc/ppp/peer/dummynet.dsl* entfernen und den Aufruf in einem Runlevel-Skript unterbringen.

Dies könnte z. B. folgendermaßen aussehen:

```
#!/bin/sh
###############################################################################
#
# /etc/init.d/dsl
#
#  Aufruf: /etc/init.d/dsl {start|stop}
#
#      Runlevel-Skript, um den DSL-Zugang zu starten
#
# Changelog
#
# 2005-06-06 Wenn /etc/rc.config nicht existiert, wird statt dessen
#            /etc/rc.status benutzt
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
```

```
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
################################################################################

### BEGIN INIT INFO
# Provides:          dsl
# Required-Start:    $local_fs procconf
# Should-Start:      pf-ipt pf-ipc dmz-ipt dmz-ipc
# Required-Stop:
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Richtet DSL ein
# Description:       Richtet DSL ein
### END INIT INFO


# Das verwendete Netzwerk-Interface

DSLIF="eth0"

# Eine kleine Routine, um zu überprüfen, ob ein Dienst gerade läuft

checkserv()
{
        ps ax | grep "$1" | grep -v grep > /dev/null
}

# Zuerst übernehmen wir all diese SuSE-Variablen, insbesondere
# rc_done und rc_failed

if test -f /etc/rc.config
then
        . /etc/rc.config
elif test -f /etc/rc.status
then
        . /etc/rc.status
fi

if test "$rc_done" == ""
then
        rc_done="\t\t\t\t\t\t\t o.k."
fi

if test "$rc_failed" == ""
then
        rc_failed="\t\t\t\t\t\t\t GING SCHIEF !!!!"
fi

# Eine Variable für den Erfolg

return=$rc_done
```

```
# Nun sehen wir mal, wie wir aufgerufen wurden

case "$1" in
start)
        echo -n "Starting pppd:"
        /sbin/ifconfig "$DSLIF" up
        /usr/sbin/pppd call dummynet.dsl 2>/dev/null
        checkserv pppd || return=$rc_failed
        echo -e "$return"
        ;;
stop)
        echo -n "Stopping pppd:"
        killall -TERM /usr/sbin/pppd 2>/dev/null
        /sbin/ifconfig "$DSLIF" down
        checkserv pppd && return=$rc_failed
        echo -e "$return"
        ;;
*)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
```

Bitte denken Sie daran, in die Variable *DSLIF* den Namen des Netzwerk-Interfaces einzutragen, das mit dem DSL-Modem verbunden ist.

Nun können Sie das Skript verlinken bzw. `insserv` aufrufen. Vorher sollten Sie es aber erst einmal von Hand mit dem Parameter `start` bzw. `stop` aufrufen, um sicherzustellen, daß sich keine Fehler in das Skript eingeschlichen haben.

Einrichten der ISDN-Karte

Die gängige Anbindung an das Internet für kleine LANs besteht heutzutage in einem ISDN-Anschluß. Prinzipiell existieren diverse Möglichkeiten, diesen zu nutzen:

Raw IP In diesem Modus werden IP-Pakete direkt ohne ein darunterliegendes Protokoll wie X.75 oder HDLC gesendet. Es findet kein Aushandeln von Parametern statt, weshalb beide Partner feste IP-Adressen besitzen müssen. Eine Identifizierung der Gegenstelle kann nur über deren Telefonnummer geschehen, da keine Anmeldung stattfindet. Obwohl dieser Modus schneller ist, wird er von den Providern normalerweise nicht unterstützt. Ich werde daher hier nicht weiter auf Raw IP eingehen.

Synchrones PPP Dieser Betriebsmodus ist der Normalfall, weswegen seine Konfiguration im folgenden genauer erklärt wird.

Asynchrones PPP Nur wenige Provider benutzen diesen Modus. Gehört Ihrer zufällig dazu, so müssen Sie Ihren Rechner konfigurieren, als ob Sie ein Modem benutzen würden. Eine genauere Beschreibung, wie dies geschieht, finden Sie in Kapitel 10, Abschnitt *Einrichten von Modem oder DSL*, ab Seite 227.

Da Sie allerdings nicht wirklich ein Modem benutzen wollen, sind ein paar Abweichungen von dem dort beschriebenen Vorgehen nötig:

- Statt einer seriellen Schnittstelle (z. B. */dev/ttyS0*) wird ein Device */dev/ttyI<n>* (z. B. /dev/ttyI0) benutzt. Dieses emuliert ein Modem und kann über AT-Befehle konfiguriert werden.
- Es müssen zusätzliche AT-Befehle in das Chat-Skript eingebaut werden:

  AT &E<*MSN/EAZ*> Mit diesem Befehl wird die MSN oder EAZ festgelegt. (Die Erklärung dieser Begriffe erfolgt weiter unten, da sie auch für »normales« ISDN von großer Bedeutung sind.)

  AT S14=<*Protocol*> Dies legt das Layer-2-Protokoll fest. Mögliche Werte sind x75i (0), x75ui (1), x75bui (2), hdlc (3). Welches Protokoll benutzt wird, müssen Sie bei Ihrem Provider erfragen.

  AT &B<*Blockgröße*> Das X.75-Protokoll benutzt verschiedene Blockgrößen. Auch hier ist es notwendig, den genauen Wert beim Provider zu erfragen.

  Weitere AT-Kommandos finden Sie in der Kernel-Dokumentation unter *<Quellenverzeichnis>/Documentation/isdn/README*.

ISDN-Modems Es gibt externe ISDN-Geräte, die sich dem Rechner gegenüber wie Modems verhalten. Aus diesem Grund entspricht auch ihre Konfiguration der ab Seite 227 beschriebenen Modemkonfiguration.

Allerdings sollten Sie zum Wählen statt des Kommandos `ATDT` oder `ATDP` lieber `ATD` verwenden. Das `T` bzw. `P` steht für Tonwahl respektive Pulswahl. Diese Unterscheidung macht aber für ISDN keinen Sinn. Außerdem benötigen Sie unter Umständen wie beim asynchronen PPP zusätzliche AT-Kommandos, um die ISDN-spezifischen Verbindungsparameter zu konfigurieren. Diese sollten Sie in der Dokumentation Ihres ISDN-Modems finden.

Benutzung eines Modems ISDN-Telefonanlagen bieten oft Analoganschlüsse, an die alte Analogtelefone angeschlossen werden können. An so einen Anschluß kann auch ein konventionelles Modem angeschlossen werden. Die Konfiguration unterscheidet sich dann allerdings in nichts von der normalen Modemkonfiguration.

Bevor Sie nun beginnen, eine Einwahl mittels ISDN zu konfigurieren, sollten Sie noch klären, wie Ihre *Multiple Subscriber Number* (MSN) oder *Endgeräte-Auswahlziffer* (EAZ) lautet. Die MSN wird bei DSS1 benutzt und entspricht normalerweise Ihrer Telefonnummer[7], die EAZ wurde bei 1TR6 benutzt und bestand nur aus einer Ziffer, die an eine Telefonnummer angehängt wurde und so eine Nebenstelle bezeichnete. MSN oder EAZ sollten in den Unterlagen stehen, die Sie nach der Einrichtung Ihres Telefonanschlusses erhalten haben.

Im folgenden werde ich nur noch von einer »MSN« sprechen. Benutzen Sie noch 1TR6, so ersetzen Sie den Begriff bitte gedanklich durch »EAZ«. Die Aufrufe zur Konfiguration dieser Nummer sind identisch.

7 Diese Angabe stimmt so nur für Deutschland. In verschiedenen europäischen Ländern existieren abweichende Arten, die MSN zu vergeben. Ein eigener Abschnitt des i4l-FAQs (*http://www.isdn4linux.de/faq*) ist diesen Fällen gewidmet.

Um nun die Einwahl mittels synchronem PPP vorzubereiten, sollten Sie als erstes ein Interface konfigurieren. Dies geschieht mit dem Befehl `isdnctrl`. Typischerweise sieht dies folgendermaßen aus:

```
/sbin/isdnctrl addif ipppO
/sbin/isdnctrl secure ipppO on
/sbin/isdnctrl eaz ipppO <MSN>
/sbin/isdnctrl encap ipppO syncppp
/sbin/isdnctrl l2_prot ipppO hdlc
/sbin/isdnctrl huptimeout ipppO 120
/sbin/isdnctrl addphone ipppO out <Provider-Nummer>
/sbin/isdnctrl dialmode ipppO auto
```

Der Befehl `isdnctrl` erhält dabei grundsätzlich als ersten Parameter eine Anweisung, gefolgt von dem Namen eines Interfaces, für das diese gilt. Schließlich folgen eventuelle weitere Angaben. Betrachten wir einmal die hier verwendeten Anweisungen:

addif Hiermit wird das Interface dem System offiziell bekanntgemacht.

secure Stellt ein, ob eingehende Einwahlversuche auf diesem Interface nur von bekannten Gegenstellen (`on`) oder von allen zugelassen werden sollen (`off`).

eaz Gibt die eigene MSN an.

encap Gibt an, ob synchrones PPP (`syncppp`) oder Raw IP (`rawip`) benutzt werden soll.

l2prot Gibt das zu verwendende Schicht-2-Protokoll an. Möglich sind unter anderem `x75i`, `x75ui`, `x75bui` und `hdlc`. Die meisten Provider verwenden `hdlc`.

huptimeout Da wir eine automatische Einwahl konfigurieren wollen, ist dies die wichtigste Einstellung überhaupt. Sie regelt, nach wie vielen Sekunden ohne Aktivität die Verbindung abgebrochen wird. Falls Sie nicht gerade einen Pauschaltarif für Ihre Internet-Anbindung bezahlen, kann Ihnen diese Einstellung bares Geld sparen.

addphone Wird hier dazu benutzt, um festzulegen, welche Telefonnummer angerufen werden soll, um eine Internet-Verbindung herzustellen (`out`). Prinzipiell können auch mehrere Aufrufe dieses Befehls dazu benutzt werden, mehrere Nummern festzulegen, die der Reihe nach probiert werden, bis eine Verbindung zustande kommt.

Mit der Angabe `in` können Nummern festgelegt werden, die ihrerseits den Rechner anrufen und so eine Verbindung aufbauen dürfen.

dialmode Dieser Befehl kontrolliert, in welcher Weise Verbindungen aufgebaut werden. `off` legt fest, daß keine Verbindungen aufgebaut und bestehende sofort geschlossen werden. `manual` verhindert zwar, daß automatisch gewählt wird, wenn auf dem Interface Netzwerkpakete gesendet werden, es kann aber mit `isdnctrl dial <Interface>` manuell gewählt werden. Ist als Wählmodus schließlich `auto` eingetragen, so wird automatisch gewählt, falls Pakete bereitliegen, aber noch keine Verbindung besteht.

Ist das Interface nunmehr eingerichtet, so muß ihm wie einem Ethernet-Device eine IP-Adresse zugeordnet werden. Zusätzlich wird auch noch festgelegt, wie die Adresse der Gegenstelle lautet. Diese Adressen sind aber normalerweise nur Platzhalter. Sie werden

später beim Verbindungsaufbau durch vom Provider übermittelte ersetzt[8]. Dies geschieht mittels `ifconfig`:

```
/sbin/ifconfig ipppO 10.1.0.1 pointopoint 10.1.0.2
```

Achten Sie aber bitte darauf, daß Sie keine Adressen benutzen, die im selben Subnetz wie die Adressen in Ihrem LAN liegen.

Nun kann das ISDN-Interface mit einer IP-Adresse angesprochen werden. Bisher werden aber nur Pakete für das Gateway 10.1.0.2 über dieses Interface geroutet. Um beliebige Rechner im Internet über dieses Interface ansprechen zu können, muß noch festgelegt werden, daß alle Pakete, die nicht für bekannte Rechner und Netze bestimmt sind, an das Gateway zu senden sind. Dies geschieht mit dem folgenden Befehl:

```
/sbin/route add default gw 10.1.0.2
```

Wird nun versucht, einen Rechner im Internet zu erreichen, so wird automatisch ein Verbindungsaufbau versucht. Dieser muß aber vorerst scheitern, da noch eine wichtige Komponente fehlt[9]. Zu Beginn einer Verbindung müssen noch wichtige Parameter ausgehandelt werden. Darunter fallen einige PPP-Optionen, Ihre tatsächliche IP-Adresse sowie Ihr Name und Paßwort.

Dies tut der Treiber aber nicht selbständig, sondern es existiert ein Systemdienst, der `ipppd`, der diese Aufgabe übernimmt. Seine Konfiguration erfolgt in mehreren Dateien. Dies sind im einzelnen:

/etc/ppp/ioptions Generelle Optionen.

/etc/ppp/peers/<name> Diese Datei erlaubt es, Optionen für einen bestimmten Verbindungspartner festzulegen. So kann man für jeden Internet-Provider eine eigene Datei mit Einstellungen definieren. Angewählt wird diese dann, indem man den `ipppd` mit dem Argument `call` *<name>* aufruft.

/etc/ppp/pap-secrets Namen und Paßwörter für die Authentisierung im Password Authentication Protocol. Diese stehen im Klartext in der Datei. Auch wird das Paßwort im Klartext an die Gegenstelle geschickt.

/ets/ppp/chap-secrets Namen und Paßwörter für die Authentisierung im Challenge Handshake Authentication Protocol. Diese stehen im Klartext in der Datei. Hierbei wird das Paßwort selbst nicht übermittelt, sondern nur benutzt, um eine Antwort auf eine zufällig gewählte Anfrage zu generieren (Challenge-Response-Verfahren). Dies ist das neuere der beiden Verfahren.

Außerdem existieren diverse Dateien, die – falls vorhanden und ausführbar – ausgeführt werden, wenn eine Verbindung auf- oder abgebaut wird. Der `ipppd` wartet allerdings nicht, bis sie beendet wurden. Für eine vollständige Liste möchte ich Sie auf die Man-

8 Es sei denn, Sie haben mit Ihrem Provider explizit feste IP-Adressen vereinbart. In diesem Fall tragen Sie bitte die vereinbarten Adressen ein.

9 Und natürlich auch, weil momentan noch keine physikalische Verbindung zwischen Ihrem Rechner und dem Telefonnetz besteht. Sie haben doch vor dem Beginn der Arbeiten das Kabel abgezogen?

page des `ipppd` verweisen (`man ipppd`). Zwei Dateien verdienen aber besondere Aufmerksamkeit:

/etc/ppp/ip-up Dieses Skript wird gestartet, wenn die Verbindung aufgebaut ist und IP-Pakete gesendet werden können.

/etc/ppp/ip-down Dies ist das Gegenstück, es wird ausgeführt, wenn keine IP-Pakete mehr gesendet werden können.

Beide Skripte werden mit den Parametern

Interface TTY Baudrate Lokale_Adresse Gateway_Adresse

aufgerufen. Dies kann man z. B. dazu benutzen, in den Dateien Firewallregeln nachzutragen, die sich auf die IP-Adresse des externen Interfaces beziehen, falls Ihnen Ihre Adresse vom Provider dynamisch zugewiesen wird.

Beginnen wir nun mit der Erstellung einer Datei */etc/ppp/ioptions*:

```
#
# /etc/ppp/ioptions
#

# ISDN-Interface
ipppO

# Dummy-Adressen
10.1.0.1:10.1.0.2

# Interface sperren
lock

# Default-Route setzen
defaultroute

# "ubergebene IP-Adressen akzeptieren
ipcp-accept-local
ipcp-accept-remote
```

Hier legen wir erst einmal diejenigen Einstellungen fest, die unabhängig vom verwendeten Provider sind. Das angegebene Interface und die IP-Adressen müssen dabei den schon getroffenen Einstellungen entsprechen. Die Option `lock` sorgt dafür, daß der `ipppd` eine spezielle Datei im Verzeichnis */var/lock* schreibt, die anderen Diensten mitteilt, daß das Device */dev/ippp0* schon verwendet wird.

Nach der Einwahl wird uns der Provider eine neue IP-Adresse zuweisen. Dies führt dazu, daß die bisher eingetragene Route, welche alle Pakete für unbekannte Ziele an das ISDN-Interface sendet, gelöscht wird. Die Option `defaultroute` sorgt dafür, daß diese Route mit der neuen Adresse des Interfaces wiederhergestellt wird.

Die letzten beiden Einstellungen sorgen schließlich dafür, daß der `ipppd` vom Provider übergebene IP-Adressen auch akzeptiert. Ohne diese Optionen würde unsere Firewall versuchen, Pakete als *10.1.0.1* an ein Gateway *10.1.0.2* zu senden. Diese Adressen sind aber im Internet nicht zulässig, womit die Pakete ignoriert würden.

Nachdem wir die Grundkonfiguration abgeschlossen haben, müssen wir noch festlegen, unter welchem Namen wir uns beim Provider anmelden wollen. Prinzipiell könnten wir auch diese Angabe in dieselbe Konfigurationsdatei schreiben, wir wollen hier aber statt dessen eine eigene providerspezifische Datei anlegen.

Eine solche Datei wird in dem Verzeichnis */etc/ppp/peers/* abgelegt. Es bietet sich an, sie nach dem Provider zu benennen. Wir wollen hier einmal annehmen, wir würden uns bei einem Provider Dummynet mit dem Benutzernamen »dummy« anmelden. Als Dateiname würde sich somit *dummynet* anbieten, hier habe ich aber noch ein »i« vor den Namen gesetzt, um deutlich zu machen, daß es hier um eine ISDN- und nicht um eine Modemeinwahl geht. Der `ipppd` ist nämlich ein jüngerer Bruder des `pppd`, der für Modemverbindungen benutzt wird. Beide benutzen daher die gleiche Art von providerspezifischen Konfigurationsdateien.

Hier also */etc/ppp/peers/idummnet*:

```
#
# /etc/ppp/peers/idummynet
#

# unser Name
name "nobody"

# Die folgenden Eintr"age dienen dem Debugging
#-detach
#debug
#kdebug 7
```

Mit der Option `name` legen wir unseren Benutzernamen fest, den wir mit dem Provider vereinbart haben. Bei T-Online ist dies z. B. eine ziemlich lange Kette von Zahlen und Sonderzeichen, die folgendermaßen aufgebaut ist:

<Anschlußkennung><T-Online-Nr.>#*<Mitbenutzernummer>*`@t-online.de`

Da diese Kennung das Zeichen # enthält, muß der Name unbedingt, wie oben dargestellt, in Anführungszeichen eingeschlossen werden. Andernfalls würde der `pppd` das # als Kommentarzeichen ansehen und es zusammen mit den nachfolgenden Zeichen ignorieren.

Um uns die eventuell notwendige Fehlersuche zu erleichtern, besteht die Möglichkeit, mit `debug` für ausführliche Meldungen zu sorgen, die alle Geschehnisse dokumentieren. Normalerweise werden diese ins Systemprotokoll geschrieben. Verbietet man dem `ipppd` aber mit der Option `-detach`, zu einem Hintergrundprozeß zu werden, so schreibt er die Meldungen auch auf den Bildschirm. Auch kann er dann durch Drücken von <Ctrl><c> beendet werden, während man dies sonst mit `kill` oder `killall` tun müßte. Für erste Versuche sind diese Einstellungen durchaus empfehlenswert.

In einigen Fällen kann es sinnvoll sein, neben den Protokollmeldungen des Programms `ipppd` weitere Meldungen des PPP-Treibers im Kernel zu erhalten. Dazu dient die Option `kdebug`. Sie erhält als Parameter einen Wert zwischen 1 und 7. 1 bedeutet hierbei, daß zusätzliche Debug-Meldungen ins Systemprotokoll geschrieben werden sollen, 2, daß alle empfangenen Pakete protokolliert werden sollen, und 4, daß alle gesendeten Pakete

gemeldet werden sollen. Diese Werte können auch addiert werden, so daß 7 die ausführlichste Protokollierung bewirkt, bei der der Inhalt eines jeden Pakets ins Systemprotokoll eingetragen wird.

Nach der Verbindungsaufnahme wird der Provider von uns verlangen, daß wir uns authentisieren. Neben dem oben festgelegten Namen brauchen wir dazu auch ein Paßwort. Es existieren zwei Methoden, mit denen der Provider feststellen kann, ob wir im Besitz des richtigen Paßworts sind. Die ältere heißt Password Authentication Protocol (PAP), die neuere Cryptographic Handshake Authentication Protocol (CHAP). Bei ersterer wird das Paßwort im Klartext übertragen, bei letzterer wird es nur dazu benutzt, eine Antwort auf eine zufällig gewählte Anfrage des Providers zu generieren.

In beiden Fällen muß dem `ipppd` das Paßwort bekannt sein, um eine Verbindung aufbauen zu können. Hierzu dienen die Dateien */etc/ppp/pap-secrets* und */etc/ppp/chap-secrets*. Beide sind identisch aufgebaut. Wenn Sie nicht wissen, welches Verfahren Ihr Provider benutzt, können Sie den gleichen Eintrag in beiden Dateien vornehmen.

Die Einträge sind dabei von der Art

Name Rechner Paßwort

Da wir den Namen des Rechners, an dem wir uns anmelden, in der Regel nicht kennen, reicht es, hier einfach »*« anzugeben. Dies könnte dann so aussehen:

```
"dummy"      *     "SeCret"
```

Hier haben wir für unseren Benutzer `dummy` das Paßwort `SeCret` eingetragen. Ich möchte aber an dieser Stelle darauf hinweisen, daß Sie dieses Paßwort selbst niemals benutzen sollten. Es ist extrem unsicher, da es

1. nur aus Buchstaben besteht,
2. in einem normalen Wörterbuch gefunden werden kann,
3. zu der Sorte Wörter gehört, die auch ein Amateur als erstes ausprobieren würde, und
4. in einem Buch als Beispiel abgedruckt ist.

Kein Paßwort, das auch nur eines dieser Kriterien erfüllt, ist für praktische Zwecke brauchbar.

Der Name und das Paßwort sind wieder in Anführungszeichen eingeschlossen. Dies ist insbesondere dann notwendig, wenn die jeweilige Zeichenkette wie im Fall einer T-Online-Kennung Sonderzeichen enthält. Es schadet aber nicht, die Anführungszeichen grundsätzlich zu verwenden.

Ein letztes Problem besteht noch in der Tatsache, daß der `ipppd`, wenn er sich beendet, die Route löscht, die dafür sorgt, daß alle Pakete für unbekannte Empfänger über das ISDN-Interface geroutet werden. Dies würde dafür sorgen, daß keine Verbindung zum Internet mehr möglich ist, sobald die Verbindung zum Provider einmal beendet wurde.

Um diesen Mißstand zu beseitigen, können wir uns der Skripte /etc/ppp/ip-up und /etc/ppp/ip-down bedienen. Sind diese schon vorhanden, so sollten wir die vorhandenen Dateien umbenennen und unsere eigenen erzeugen:

```
# cd /etc/ppp
# mv ip-up ip-up.orig
# mv ip-down ip-down.orig
# echo '#!/bin/sh' > ip-up
# echo '#!/bin/sh' > ip-down
# chmod 700 ip-up ip-down
```

In diesem Fall interessiert uns besonders ip-down, da es ausgeführt wird, wenn die Verbindung abgebaut wurde. Es erhält als einen Parameter die gegenwärtige IP-Adresse des ISDN-Interfaces. Es braucht nur eine neue Route zu dieser Adresse zu setzen, und unser Problem ist gelöst:

```
#!/bin/sh
#
# /etc/ppp/ip-down
#

defgw=$5
logger ip-down: Neues Gateway $defgw
/sbin/route add default gw $defgw
```

Hier wird der Vorgang auch noch im Systemprotokoll vermerkt. Wenn dies nicht gewünscht wird, kann der logger-Befehl auskommentiert werden.

Nun sind wir fast fertig. Allerdings ist es doch etwas unpraktisch, die ganzen isdnctrl-Befehle von Hand einzugeben. Aus diesem Grunde hier ein kleines Runlevel-Skript:

```
#!/bin/sh
########################################################################
#
# /etc/init.d/isdn
#
#  Aufruf: /etc/init.d/isdn {start|stop}
#
#      Runlevel-Skript, um den ISDN-Zugang zu starten
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder h"ohere Version
#
########################################################################

### BEGIN INIT INFO
# Provides:          isdn
# Required-Start:    $local_fs procconf
# Should-Start:      pf-ipt pf-ipc dmz-ipt dmz-ipc
# Required-Stop:
# Should-Stop:
# Default-Start:     2 3 5
# Default-Stop:      0 1 6
# Short-Description: Richtet ISDN ein
# Description:       Richtet ISDN ein
### END INIT INFO
```

```
# Unsere eigene Telefonnummer, MSN, EAZ, ...
msn=7654321

# Die Nummer unseres Providers
provider=123456

# Dateiname der Provideroptionen
optfile=idummynet

case $1 in
start)
        echo Starte ISDN ...

        /sbin/isdnctrl addif ipppO
        /sbin/isdnctrl secure ipppO on
        /sbin/isdnctrl eaz ipppO $msn
        /sbin/isdnctrl encap ipppO syncppp
        /sbin/isdnctrl l2_prot ipppO hdlc
        /sbin/isdnctrl huptimeout ipppO 120
        /sbin/isdnctrl addphone ipppO out $provider
        /sbin/isdnctrl dialmode ipppO auto
        /sbin/ifconfig ipppO 10.1.0.1 pointopoint 10.1.0.2
        /sbin/route add default gw 10.1.0.2
        /sbin/ipppd call $optfile
        ;;

stop)
        echo Beende ISDN ...

        /sbin/isdnctrl hangup ipppO
        /usr/bin/killall ipppd
        /sbin/ifconfig ipppO down
        ;;

*)
        echo "Usage: isdn {start|stop}"
esac
```

Die hier konfigurierten Telefonnummern müssen Sie an Ihre Bedürfnisse anpassen. Für erste Experimente habe ich im Anhang A ab Seite 595 eine Liste mit Providern zusammengestellt, die keine Anmeldung verlangen und über die Telefonrechnung abrechnen.

Nun könnten wir im Prinzip beginnen, mit dem Internet Kontakt aufzunehmen. Dies wäre aber nicht ratsam, da wir noch kein Firewalling eingerichtet haben. Auch funktioniert die Namensauflösung noch nicht, so daß wir Rechner nur unter ihren IP-Adressen ansprechen können.

Lesen Sie daher bitte weiter, bis Sie in Kapitel 10, Abschnitt *Verbindungstests*, ab Seite 257 erfahren, wie Sie Ihre ISDN-Anbindung testen können. Bevor Sie das dort beschriebene Skript ausführen, vergewissern Sie sich bitte, daß der `ipppd` noch nicht gestartet und der Rechner nicht mit der Telefondose verbunden ist. Nun starten Sie wie beschrieben das Skript.

Wenn Sie an die Stelle kommen, an der Sie aufgefordert werden, die Verbindung zum Internet herzustellen, stellen Sie bitte die Verbindung zwischen ISDN-Karte und Telefonan-

lage her, wechseln in eine andere Konsole, und rufen Sie das oben dargestellte Runlevel-Skript mit dem Parameter »start« auf.

Wenn Sie aufgefordert werden, die Verbindung zu beenden, so rufen Sie das Skript bitte mit »stop« auf und ziehen das Telefonkabel aus der ISDN-Karte oder der Telefondose.

Wenn alles wie gewünscht funktioniert, können Sie das Skript verlinken bzw. `insserv` aufrufen.

Eintrag des DNS-Servers

Bis jetzt müssen die Maschinen im lokalen Netz IP-Adressen (z. B. *192.168.3.110*) benutzen, da kein Mechanismus vorhanden ist, diese Adressen in symbolische Namen (z. B. *host110.dummy.local*) umzusetzen[10]. Im Internet existieren für diese Zwecke Server, die gewissermaßen als digitale Telefonbücher dienen und diese Umsetzung erledigen.

Um diese nutzen zu können, ist es nötig, dem System mitzuteilen, wie sie erreicht werden können. Dazu muß die IP-Adresse des gewünschten DNS-Servers in die Datei */etc/resolv.conf* eingetragen werden. Neben der Adresse des DNS-Servers kann dort auch eingetragen werden, welche Domänen an eine Adresse angehängt werden sollen, wenn nur der Rechnername angegeben wurde.

Sollen z. B. die DNS-Server *10.0.0.77* und *10.0.0.88* benutzt werden und soll es möglich sein, Adressen in den Domänen *dummy1.local* und *dummy2.local* nur als Rechnernamen anzugeben, so könnte die Datei *resolv.conf* so aussehen:

```
# /etc/resolv.conf
#
# Beispielversion, bitte an die eigenen Bedürfnisse anpassen!
#
search dummy1.local dummy2.local
nameserver 10.0.0.77
nameserver 10.0.0.88
```

Wenn wir die DNS-Server unseres Providers nicht kennen, hilft ein kleiner Trick. Grundsätzlich teilen uns viele Provider bei der Einwahl mit, welche DNS-Server wir benutzen sollen.

Wenn wir den Parameter

```
usepeerdns
```

in die Provideroptionen-Datei

```
/etc/ppp/peers/<Providername>
```

eintragen, erzeugt der `pppd` bei der Einwahl eine Datei */etc/ppp/resolv.conf*, welche die von uns benötigten `nameserver`-Zeilen enthält. Diese können wir dann später um die `search`-Zeile erweitern und als */etc/resolv.conf* verwenden.

10 Abgesehen von der Datei */etc/hosts*, in der solche Zuordnungen eingetragen werden können. Dies aber für alle Rechner im Internet zu tun, ist natürlich nicht praktikabel.

Bis dahin benötigen wir aber eine vorläufige *resolv.conf*-Datei. In dieser sollte ein DNS-Server eingetragen sein, der eine Adresse besitzt, die außerhalb des lokalen Netzes liegt. Dazu bietet sich die Dummy-Adresse an, die wir für das Einwahl-Gateway unseres Providers angegeben haben. Diese Datei speichern wir als */etc/ppp/resolv.conf*. Anschließend verlinken wir sie dann als */etc/resolv.conf*.

In unserem Beispiel geschieht das mit den folgenden Befehlen:

```
# cd /etc
# mv resolv.conf resolv.conf.orig
# echo "nameserver 10.1.0.2" > ppp/resolv.conf
# ln -s ppp/resolv.conf resolv.conf
```

Wenn wir nun die Verbindungstests[11] durchführen, wird der erste Test mißlingen, da *10.1.0.2* kein gültiger Nameserver ist. Der Versuch, eine Anfrage an ihn zu stellen, sollte aber dazu führen, daß eine Verbindung aufgebaut wird. Dadurch wird unsere Dummy-*resolv.conf* durch eine Version mit gültigen DNS-Servern ersetzt.

Machen wir nun einen weiteren Versuch, so werden gültige DNS-Server verwendet, und die Tests gelingen. Wir können dies auch abkürzen, indem wir dem Testskript mehrere Namen von Zielrechnern als Parameter mitgeben oder den gleichen Namen mehrfach verwenden. Dann sollte nur der Versuch, den Namen des ersten Rechners aufzulösen, mißlingen. Bei den Namen der anderen angegebenen Rechner sollte alles glattgehen.

Sind die Verbindungstests abgeschlossen, sollten wir aber die search-Zeile eintragen und den Link wieder durch eine richtige Datei ersetzen:

```
# cd /etc
# rm resolv.conv
# echo ßearch dummy1.local dummy2.local" > resolv.conf
# cat ppp/resolv.conf >> resolv.conf
```

Ohne diese Änderung könnten wir später Probleme mit dem Firewalling bekommen, falls der Provider, z. B. um die Last besser zu verteilen, bei jeder Einwahl andere DNS-Server angibt. Das Skript erlaubt nur den Zugriff auf die in der Datei */etc/resolv.conf* eingetragenen DNS-Server.

Die Datei wird aber vor dem Start der Verbindung ausgelesen. Ändert sie sich beim Verbindungsaufbau, so führt dies dazu, daß die Anwendungen DNS-Server benutzen, die im Firewalling nicht freigeschaltet sind. Alle Verbindungsversuche enden dann mit »Host not found«.

11 Vgl. Kapitel 10, Abschnitt *Verbindungstests*, ab Seite 257

Eintragen der lokalen Rechnernamen

Nun, da ein DNS-Server eingetragen ist, kann es passieren, daß diverse Programme in ihren Ausgaben versuchen, numerische Adressen in logische umzuwandeln. Ein Beispiel ist der Aufruf

```
route
```

der die Routing-Tabelle anzeigt. Dem aufmerksamen Beobachter wird auffallen, daß vor der Anzeige von numerischen Adressen merkliche Pausen entstehen. Dies rührt daher, daß der eingetragene DNS-Server gefragt wird, wozu jedesmal erst eine Internet-Verbindung aufgebaut wird.

Die beste Lösung für dieses Problem wäre sicherlich, einen eigenen DNS-Server für das lokale Netz aufzubauen. Alternativ können aber auch die Dateien */etc/hosts* und */etc/networks* benutzt werden.

In einem lokalen Netz mit zwei Rechnern, welche die Adressen 192.168.20.15 und 192.168.20.100 haben, die Namen win1 und fw1 besitzen und sich im Netz dummy1.local befinden, könnte die Datei */etc/hosts* folgendermaßen aussehen:

```
127.0.0.1       localhost
192.168.20.100  win1.dummy1.local  win1
192.168.20.15   fw1.dummy1.local   fw1
```

Wie wir sehen, werden einer numerischen Adresse eine oder mehrere logische Bezeichnungen zugewiesen, die dann anstelle der tatsächlichen Adresse verwendet werden können.

Eine derartige Zuweisung ist auch für Netzwerknamen möglich. Hierzu dient die Datei */etc/networks*, die in unserem Beispiel folgendermaßen aussehen könnte:

```
loopback      127.0.0.0
dummy1.local  192.168.20.0
```

Um sicherzustellen, daß auch tatsächlich zuerst in den Konfigurationsdateien nachgesehen wird, bevor eine DNS-Anfrage durchgeführt wird, muß je nach Version und Distribution entweder

```
order hosts, bind
```

in der Datei */etc/host.conf* oder

```
hosts:    files dns
networks: files dns
```

in der Datei */etc/nsswitch.conf* eingetragen sein. Dies ist allerdings normalerweise der Fall.

Verbindungstests

Idealerweise würde man nun ein eigenes Testnetz aufbauen, um zu testen, ob die Konfiguration der Netzwerkschnittstellen auch funktioniert hat. Für den Test der Netzwerkkarten reicht es dabei, einen zusätzlichen alleinstehenden Rechner mit der jeweils zu testenden Netzwerkkarte zu verbinden. Nachdem man ihm eine passende IP-Adresse gegeben hat, sollte ein einfaches

```
# ping <IP-Adresse des Prüfrechners>
```

ausreichen, um festzustellen, ob die Netzwerkkarte richtig konfiguriert ist.

Der Test von Modems und ISDN-Karten gestaltet sich da schon aufwendiger. Um hier ein Testnetz aufzubauen, ist ein weiterer Rechner nötig, der ebenfalls ein Modem oder eine ISDN-Karte besitzt, sowie eine Telefonanlage zur Verbindung der beiden. Es muß auch eine Software auf dem Prüfrechner installiert sein, die den Verbindungsaufbau durch den Provider simuliert. Dies bedeutet, daß man auch auf der Gegenseite einen `pppd` konfigurieren muß, so daß dieser die gleichen Authentisierungsmaßnahmen einsetzt wie der Provider. Dies ist sinnvoll, wenn man häufiger Firewalls aufsetzen muß. Für eine einzelne Installation verdoppelt man nur die Möglichkeiten, Fehler zu begehen, die dann erst langwierig gesucht werden müssen.

Einfacher ist der Test, wenn wir uns überwinden und doch eine Verbindung zum Internet aufbauen. Da dies aber nicht ungeschützt geschehen sollte, müssen wir vorher einige einfache Firewallregeln definieren, die nur die von uns gewünschten Anfragen erlauben, die aber jeglichen sonstigen Verkehr verhindern. Die hierfür benötigten Firewallregeln sind glücklicherweise einfach genug, daß kaum die Gefahr besteht, etwas falsch zu machen.

Sind die Firewallregeln aktiviert, können wir relativ gefahrlos eine Verbindung zum Internet aufbauen und testen, ob wir externe Rechner erreichen. Dabei werden wir aber hier darauf verzichten, Ping-Anfragen abzusetzen. Statt dessen werden wir im folgenden nur versuchen, Rechnernamen aufzulösen. Da hierzu eine Verbindung zu einem DNS-Server im Internet geöffnet werden muß, reicht dieser Test aus, um festzustellen, ob bei der Konfiguration des Internet-Zugangs Fehler gemacht wurden.

Dies soll aber nicht bedeuten, daß Ping-Anfragen prinzipiell zu unsicher sind. Vielmehr besteht das Problem darin, daß immer weniger Rechner auf Ping-Anfragen reagieren. Wenn wir also nun versuchen, einen Rechner mit `ping` anzusprechen, und es geht schief, so wissen wir nicht, ob es an der Netzwerkkonfiguration liegt oder ob der Administrator des Zielrechners besonders sicherheitsbewußt ist. DNS-Abfragen gültiger Rechnernamen sollten aber funktionieren, da ansonsten ein normales Surfen im Internet unmöglich ist.

Im folgenden werden wir sehen, wie ein solcher Test ablaufen kann. Da einige Unterschiede zwischen dem Firewalling mit `ipchains` (Kernel 2.2.x) und dem mit `iptables` (Kernel 2.4.x) existieren, erscheint der nächste Abschnitt doppelt. Es genügt, wenn Sie denjenigen lesen, der für Ihr System paßt.

Mit ipchains (Kernel 2.2.x)

Um für unseren Test nicht immer wieder manuell dieselben Befehle eingeben zu müssen, hier ein kleines Skript, das den Vorgang etwas erleichtert:

```
#!/bin/sh
# ********************************************************************
# ECHOLOT.SH
#
# Dieses Skript versucht, einen fremden Rechner zu erreichen,
# ohne dabei die Schilde so weit herunterzulassen, daß der
# Rechner angreifbar wäre
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
# ********************************************************************

# Startmeldung

echo
echo "**** Echolot v0.3.1 --- Test der Netzwerkanbindung ****"

# Ein paar Variablen ...

TESTHOSTS="www.oreilly.de"

if test "x$*" != "x"
then
        TESTHOSTS="$*"
fi

# Bau eines Dateinamens aus Datum, Uhrzeit, gegenwärtiger
# Prozeßnummer und Zufallsbytes aus /dev/urandom

tmpsuggestion(){
        SNAME=tmp-‘date +%Y%d%m%k%M%S‘-"$$"
        SNAME="$SNAME"-‘dd if=/dev/urandom \
        bs=1 count=16 |\
        cat -A| tr -d -c  [:alnum:]|tr -s [:alnum:]‘
        echo "$SNAME"
}
```

```
# Überprüfung, ob eine Datei dieses Namens schon existiert

NEWNAME=‘tmpsuggestion‘

while test -e "$NEWNAME"
do
        NEWNAME=‘tmpsuggestion‘
done

# dummy

echo
echo Temporäre Datei: "$NEWNAME"

# Sicherung der gegenwärtigen Firewallregeln

echo
echo Sichern der Firewallregeln:

/sbin/ipchains-save > "$NEWNAME"

# Neue Regeln

# - alles abschotten
/sbin/ipchains -P input DENY
/sbin/ipchains -P forward DENY
/sbin/ipchains -P output DENY

/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward

# - DNS erlauben

/sbin/ipchains -A input  -p UDP --sport 53 -j ACCEPT
/sbin/ipchains -A output -p UDP --dport 53 -j ACCEPT

/sbin/ipchains -A input  -p TCP --sport 53 ! -y -j ACCEPT
/sbin/ipchains -A output -p TCP --dport 53 -j ACCEPT

# Der Rechner sollte sicher sein. Der Benutzer kann also die Verbindung
# zum Internet herstellen.

echo
echo "------------------------------------------------------------------"
echo Bitte verbinden Sie den Rechner nun mit dem Internet, und drücken Sie
echo "<Return>"
echo "------------------------------------------------------------------"
echo

read dummy

# Die Verbindung sollte nun bestehen. Testen wir sie.

echo
echo ’*** Teste Namensauflösung ****’
```

```
for TESTHOST in  $TESTHOSTS
do

   echo
   sleep 3
   RESULT=‘dig "$TESTHOST" | grep "$TESTHOST" \
   | grep -v ’^;’‘
   if test "$RESULT" == ""
   then
      echo "$TESTHOST konnte nicht aufgelöst werden!"
   else
      echo  "$RESULT"
fi

done


# Bevor wir den Ursprungszustand wiederherstellen, sollte die Verbindung zum
# Internet gekappt werden.

echo
echo "------------------------------------------------------------"
echo Bitte trennen Sie den Rechner nun vom Internet, und drücken Sie
echo "<Return>"
echo "------------------------------------------------------------"
echo

read dummy

# Wiederherstellung des Anfangszustandes

echo Wiederherstellung des Ausgangszustandes ...
/sbin/ipchains -F
/sbin/ipchains-restore < "$NEWNAME"

echo
rm -i "$NEWNAME"
```

Auf die Befehle `ipchains`, `ipchains-save` und `ipchains-restore` komme ich in Kapitel 11 ab Seite 269 noch zu sprechen. Sie dienen zur Festlegung, Sicherung und Wiederherstellung von Filterregeln.

Führt man die oben wiedergegebene Datei mit Rootrechten aus, so erhält man in etwa folgende Meldungen:

```
**** Echolot v0.3.1 --- Test der Netzwerkanbindung ****

Temporäre Datei: tmp-20000211212232-4957-MOMFMLMtdMkJtMAMsr

Sichern der Firewallregeln:
Saving ‘input’.

------------------------------------------------------------------
Bitte verbinden Sie den Rechner nun mit dem Internet, und drücken Sie
<Return>
------------------------------------------------------------------
```

Das Skript hat nun die gegenwärtig aktiven Firewallregeln in eine temporäre Datei namens *tmp-20000211212232-4957-MOMFMLMtdMkJtMAMsr* im aktuellen Verzeichnis gesichert und neue Filterregeln installiert. Man sollte dieses Skript nur in einem Verzeichnis ausführen, auf das ausschließlich `root` Schreibzugriff hat, da ansonsten jeder Benutzer mit Schreibrecht auf dem Verzeichnis und ein bißchen Phantasie symbolische Links anlegen kann, die dazu führen, daß beliebige Dateien überschrieben werden.

Der Rechner ist nun durch Filterregeln gesichert. Wir können also die physikalische Verbindung zum zu testenden Netzwerk herstellen (z. B. das Modem mit der Telefondose verbinden). Muß der zu testende Netzwerkdienst (z. B. `pppd`) erst manuell gestartet werden, so sollten wir auch das tun (gegebenenfalls in einer eigenen Konsole mit <Alt><F_x>).

Nun können wir mit <Return> den eigentlichen Test in Gang setzen. Der Rechner versucht dabei, die IP-Adressen zu bestimmten logischen Rechnernamen zu ermitteln. Wurden dem Skript Rechnernamen auf der Kommandozeile mitgegeben (z. B. `./echolot.sh www.heise.de www.oreilly.de`), so benutzt er diese, andernfalls die Rechner, die im Skript in der Variablen *TESTHOSTS* angegeben sind.

Hier waren *www.heise.de* und *www.oreilly.de* als Ziel vorgegeben:

```
*** Teste Namensauflösung ****

www.heise.de.           16h14m8s IN A   193.99.144.71

www.oreilly.de.         11h31m32s IN CNAME  orade.oreilly.de.
```

Wir sehen, daß der Rechner *www.heise.de* die IP-Adresse *193.99.144.71* besitzt, während *www.oreilly.de* ein anderer Name für *orade.oreilly.de* ist.

Gelingt die Namensauflösung dagegen nicht, so wird folgende Meldung ausgegeben:

```
*** Teste Namensauflösung ****

www.nowhere.nodomain konnte nicht aufgelöst werden!
```

In diesem Fall existiert der Rechner *www.nowhere.nodomain* nicht. Es ist daher nicht überraschend, daß eine Namensauflösung nicht gelang. Wenn man aber mehrere Adressen bekannter Rechner (z. B. Webserver) ausprobiert, ohne ein einziges Mal erfolgreich zu sein, so liegt die Vermutung nahe, daß ein Problem bei der Konfiguration der Internet-Anbindung vorliegt. Dies sollten Sie insbesondere auch dann annehmen, wenn Sie Meldungen wie die folgenden sehen:

```
*** Teste Namensauflösung ****

;; res_nsend to server default -- 62.104.196.134: Connection refused
www.heise.de konnte nicht aufgelöst werden!

;; res_nsend to server default -- 62.104.196.134: Connection refused
www.oreilly.de konnte nicht aufgelöst werden!
```

In diesem Fall hatte ich vor dem Start des Testes den pppd beendet, so daß die DNS-Anfragen nicht zugestellt werden konnten. In einem anderen Fall hatte ich meine Anfrage gestartet, bevor der pppd die Verbindung zum Provider hergestellt hatte:

```
*** Teste Namensauflösung ****

;; res_nsend to server default -- 62.104.196.134: Connection timed out
www.heise.de konnte nicht aufgelöst werden!

www.oreilly.de.         11h31m59s IN CNAME  orade.oreilly.de.
```

Man sieht, wie hier die erste Auflösung mißlingt, die zweite aber erfolgreich ist.

Vermutet man nun ein Problem mit der Netzwerkkonfiguration, so liefern die Aufrufe ipconfig und route -n erste Anhaltspunkte. Der erste sollte Ihr externes Interface anzeigen (hier: ppp0):

```
> /sbin/ifconfig
[...]
ppp0  Link encap:Point-to-Point Protocol
      inet addr:62.180.197.128  P-t-P:10.1.0.2  Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:16804 errors:1427 dropped:0 overruns:0 frame:1427
      TX packets:13213 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:10
```

Ist dies nicht der Fall, so ist Ihr Netzwerk-Interface nicht konfiguriert bzw. Ihr pppd oder ipppd ist nicht gestartet. Hier sollte man die Systemprotokolle untersuchen, um dort Hinweise zu finden, was nicht nach Plan gelaufen ist. Man findet die Protokolldateien normalerweise unter */var/log*. Die wichtigste Datei heißt *messages*, aber manche Fehlermeldungen werden auch in andere Dateien geschrieben (siehe Kapitel 9, Abschnitt *Das Systemprotokoll*, ab Seite 205).

Mit route kann man feststellen, über welches Interface Pakete zugestellt werden:

```
> /sbin/route -n
Kernel IP routing table
Destination  Gateway   Genmask         Flags Metric Ref  Use Iface
10.1.0.2     0.0.0.0   255.255.255.255 UH    0      0      0 ppp0
127.0.0.0    0.0.0.0   255.0.0.0       U     0      0      0 lo
0.0.0.0      10.1.0.2  0.0.0.0         UG    0      0      0 ppp0
```

Hier sollten normalerweise zwei Einträge für Ihr externes Interface vorhanden sein. In diesem Fall erlaubt es die erste Regel, gezielt das Gateway des Providers anzusprechen. So eine Regel wird normalerweise automatisch eingerichtet, wenn wir ein Netzwerk-Interface aktivieren oder den pppd bzw. ipppd starten.

Mit ihr können wir aber noch nicht Pakete an beliebige Rechner im Internet senden. Dazu dient die dritte Regel. Sie besagt: »Pakete für Rechner, deren Adressen nicht durch andere Regeln abgedeckt sind« (Destination = *0.0.0.0*, Genmask = *0.0.0.0*), »an den Provider schicken« (Gateway = *10.1.0.2*, Iface = ppp0). Um diese Regel zu erzeugen, muß im Falle des pppd die Option defaultroute in der Konfigurationsdatei angegeben werden,

während beim ipppd oder der Verwendung einer Netzwerkkarte ein passender Aufruf von route im Startup-Skript stehen muß.

Hat das Skript versucht, alle eingetragenen Rechner zu erreichen, so gibt es die folgende Meldung aus:

```
------------------------------------------------------------
Bitte trennen Sie den Rechner nun vom Internet, und drücken Sie
<Return>
------------------------------------------------------------
```

Nun sollten alle Netzwerkverbindungen getrennt werden, indem die entsprechenden Kabel von den jeweiligen Anschlußdosen abgezogen werden. Manuell gestartete Dienste sollten beendet werden.

Nach dem Betätigen von <Return> räumt das Skript noch etwas auf:

```
Wiederherstellung des Ausgangszustandes ...

rm: »tmp-20000211212232-4957-MOMFMLMtdMkJtMAMsr« entfernen? j
```

Die alten Filterregeln sind nun wiederhergestellt. Falls der Benutzer es wünscht, wird die temporäre Datei mit dem Ausgangszustand des Systems gelöscht.

Mit iptables (Kernel 2.4.x)

Auch für iptables habe ich ein kleines Skript geschrieben, das den Vorgang etwas erleichtert:

```
#!/bin/sh
# ******************************************************************************
# ECHOLOT.SH
#
# Dieses Skript versucht, einen fremden Rechner zu erreichen,
# ohne dabei die Schilde so weit herunterzulassen, daß der
# Rechner angreifbar w"are.
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
# ******************************************************************************
```

```
# ------------------------------------------------------------------------
# Startmeldung
# ------------------------------------------------------------------------

echo
echo "**** Echolot v0.4.1 (iptables) --- Test der Netzwerkanbindung ****"

# ------------------------------------------------------------------------
# Variablen
# ------------------------------------------------------------------------

# Zielsystem

TESTHOSTS="www.oreilly.de"

if test "x$*" != "x"
then
    TESTHOSTS="$*"
fi

# Eine Kurzbezeichnung für ipchains

R=/sbin/iptables

# ------------------------------------------------------------------------
# Funktionen
# ------------------------------------------------------------------------

# closeall()
#     alle Verbindungen verbieten

closeall(){


#   Alle Regeln löschen

    $R -F INPUT
    $R -F FORWARD
    $R -F OUTPUT

#   Alle Pakete, die nicht explizit erlaubt sind, sind verboten

    $R -P INPUT DROP
    $R -P FORWARD DROP
    $R -P OUTPUT DROP

}

# allowdns()
#    DNS erlauben

allowdns(){

    $R -A INPUT  -p UDP --sport 53 -j ACCEPT
    $R -A OUTPUT -p UDP --dport 53 -j ACCEPT

    $R -A INPUT  -p TCP --sport 53 ! --syn -j ACCEPT
    $R -A OUTPUT -p TCP --dport 53 -j ACCEPT
}
```

```
# allowping()
#    pings zulassen

allowping(){

    $R -A INPUT  -p icmp --icmp-type 0 -j ACCEPT
    $R -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT
}

# testcon(<host>)
#    kann die IP-Adresse zu <host> ermittelt werden?

testcon(){

    RESULT=‘dig "$TESTHOST" | grep "$TESTHOST" \
    | grep -v ’^;’‘
    if test "$RESULT" == ""
    then
        echo "$TESTHOST konnte nicht aufgelöst werden!"
    else
        echo  "$RESULT"
    fi

}


# ------------------------------------------------------------------------
# Hauptprogramm
# ------------------------------------------------------------------------

# Paketfilter konfigurieren

closeall
allowdns
# allowping

# Der Rechner sollte sicher sein. Der Benutzer kann also die Verbindung
# zum Internet herstellen.


echo
echo "------------------------------------------------------------------"
echo Bitte verbinden Sie den Rechner nun mit dem Internet, und drücken Sie
echo "<Return>"
echo "------------------------------------------------------------------"
echo

read dummy

# Die Verbindung sollte nun bestehen. Testen wir sie.

echo
echo ’*** Löse die Namen der Zielrechner auf ****’

for TESTHOST in  $TESTHOSTS
do

    testcon "$TESTHOST"

done
```

```
# Nun sollte die Verbindung zum Internet gekappt werden.

echo
echo "------------------------------------------------------------"
echo Bitte trennen Sie den Rechner nun vom Internet, und drücken Sie
echo "<Return>"
echo "------------------------------------------------------------"
echo

read dummy

# Alle Netzwerkverbindungen verhindern

closeall

echo
echo "============================================================"
echo "Test beendet, Netzwerkverbindungen sind nicht mehr möglich"
echo "============================================================"
echo
```

Führt man die oben wiedergegebene Datei mit Rootrechten aus, so erhält man in etwa folgende Meldungen:

```
**** Echolot v0.4.1 (iptables) --- Test der Netzwerkanbindung ****

---------------------------------------------------------------------
Bitte verbinden Sie den Rechner nun mit dem Internet, und drücken Sie
<Return>
---------------------------------------------------------------------
```

Der Rechner wird nun durch Filterregeln gesichert. Wir können also die physikalische Verbindung zum zu testenden Netzwerk herstellen (z. B. das Modem mit der Telefondose verbinden). Muß der zu testende Netzwerkdienst (z. B. `pppd`) erst manuell gestartet werden, so sollten wir auch das tun (gegebenenfalls in einer eigenen Konsole mit <Alt><F_x>).

Nun können wir mit <Return> den eigentlichen Test in Gang setzen. Der Rechner versucht dabei, die IP-Adressen zu bestimmten logischen Rechnernamen zu ermitteln. Wurden dem Skript Rechnernamen auf der Kommandozeile mitgegeben (z. B. `./echolot.sh www.heise.de www.oreilly.de`), so benutzt er diese, andernfalls die Rechner, die im Skript in der Variablen *TESTHOSTS* angegeben sind.

Hier waren *www.heise.de* und *www.oreilly.de* als Ziel vorgegeben:

```
*** Teste Namensauflösung ****

www.heise.de.           16h14m8s IN A   193.99.144.71

www.oreilly.de.         11h31m32s IN CNAME  orade.oreilly.de.
```

Wir sehen, daß der Rechner *www.heise.de* die IP-Adresse *193.99.144.71* besitzt, während *www.oreilly.de* ein anderer Name für *orade.oreilly.de* ist.

Gelingt die Namensauflösung dagegen nicht, so wird folgende Meldung ausgegeben:

```
*** Teste Namensauflösung ****

www.nowhere.nodomain konnte nicht aufgelöst werden!
```

In diesem Fall existiert der Rechner *www.nowhere.nodomain* nicht. Es ist daher nicht überraschend, daß eine Namensauflösung nicht gelang. Wenn man aber mehrere Adressen bekannter Rechner (z. B. Webserver) ausprobiert, ohne ein einziges Mal erfolgreich zu sein, so liegt die Vermutung nahe, daß ein Problem bei der Konfiguration der Internet-Anbindung vorliegt. Dies sollten Sie insbesondere auch dann annehmen, wenn Sie Meldungen wie die folgenden sehen:

```
*** Teste Namensauflösung ****

;; res_nsend to server default -- 62.104.196.134: Connection refused
www.heise.de konnte nicht aufgelöst werden!

;; res_nsend to server default -- 62.104.196.134: Connection refused
www.oreilly.de konnte nicht aufgelöst werden!
```

In diesem Fall hatte ich vor dem Start des Testes den `pppd` beendet, so daß die DNS-Anfragen nicht zugestellt werden konnten. In einem anderen Fall hatte ich meine Anfrage gestartet, bevor der `pppd` die Verbindung zum Provider hergestellt hatte:

```
*** Teste Namensauflösung ****

;; res_nsend to server default -- 62.104.196.134: Connection timed out
www.heise.de konnte nicht aufgelöst werden!

www.oreilly.de.         11h31m59s IN CNAME  orade.oreilly.de.
```

Man sieht, wie hier die erste Auflösung mißlingt, die zweite aber erfolgreich ist.

Vermutet man nun ein Problem mit der Netzwerkkonfiguration, so liefern die Aufrufe `ipconfig` und `route -n` erste Anhaltspunkte. Der erste sollte Ihr externes Interface anzeigen (hier: `ppp0`):

```
> /sbin/ifconfig
[...]
ppp0  Link encap:Point-to-Point Protocol
      inet addr:62.180.197.128  P-t-P:10.1.0.2  Mask:255.255.255.255
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:16804 errors:1427 dropped:0 overruns:0 frame:1427
      TX packets:13213 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:10
```

Ist dies nicht der Fall, so ist Ihr Netzwerk-Interface nicht konfiguriert bzw. Ihr `pppd` oder `ipppd` ist nicht gestartet. Hier sollte man die Systemprotokolle untersuchen, um dort Hinweise zu finden, was nicht nach Plan gelaufen ist. Man findet die Protokolldateien normalerweise unter */var/log*. Die wichtigste Datei heißt *messages*, aber manche Fehlermeldungen werden auch in andere Dateien geschrieben (siehe Kapitel 9, Abschnitt *Das Systemprotokoll*, ab Seite 205).

Mit route kann man feststellen, über welches Interface Pakete zugestellt werden:

```
> /sbin/route -n
Kernel IP routing table
Destination  Gateway   Genmask         Flags Metric Ref  Use Iface
10.1.0.2     0.0.0.0   255.255.255.255 UH    0      0      0 ppp0
127.0.0.0    0.0.0.0   255.0.0.0       U     0      0      0 lo
0.0.0.0      10.1.0.2  0.0.0.0         UG    0      0      0 ppp0
```

Hier sollten normalerweise zwei Einträge für Ihr externes Interface vorhanden sein. In diesem Fall erlaubt es die erste Regel, gezielt das Gateway des Providers anzusprechen. So eine Regel wird normalerweise automatisch eingerichtet, wenn wir ein Netzwerk-Interface aktivieren oder den pppd bzw. ipppd starten.

Mit ihr können wir aber noch nicht Pakete an beliebige Rechner im Internet senden. Dazu dient die dritte Regel. Sie besagt: »Pakete für Rechner, deren Adressen nicht durch andere Regeln abgedeckt sind« (Destination = *0.0.0.0*, Genmask = *0.0.0.0*), »an den Provider schicken« (Gate way = *10.1.0.2*, Iface = ppp0). Um diese zu erzeugen, muß im Falle des pppd die Option defaultroute in der Konfigurationsdatei angegeben werden, während beim ipppd oder der Verwendung einer Netzwerkkarte ein passender Aufruf von route im Startup-Skript stehen muß.

Hat das Skript versucht, alle eingetragenen Rechner zu erreichen, so gibt es die folgende Meldung aus:

```
------------------------------------------------------------
Bitte trennen Sie den Rechner nun vom Internet, und drücken Sie
<Return>
------------------------------------------------------------
```

Nun sollten alle Netzwerkverbindungen getrennt werden, indem die entsprechenden Kabel von den jeweiligen Anschlußdosen abgezogen werden. Manuell gestartete Dienste sollten beendet werden.

Nach dem Betätigen von <Return> werden Firewallregeln definiert, die jeglichen Netzwerkverkehr unterbinden. Das System ist nun sicher. Nicht zum Surfen nutzbar, aber sicher vor Netzwerkangriffen.