Deutsches Linux NET-2/3-HOWTO

Übersetzung durch Peter Sütterlin (pit@uni-sw.gwdg.de) v3.5d, August 1996

Ziel dieses Textes ist die Beschreibung, wie man die NET-2 und NET-3 Software für Linux bekommt und wie sie installiert und konfiguriert wird. Einige Antworten auf öfters gestellte Fragen sind im Anhang enthalten. Dieses Dokument ist nicht dazu gedacht, dem Leser eine Einführung in das TCP/IP Netzprotokoll zu geben, auch wenn diesbetreffende Informationen wo immer möglich mitaufgenommen wurden. Eine Liste mit Texten, die eine solche TCP/IP-Einführung aufweisen, ist aber angegeben. Der Text ist eine deutsche Übersetzung des englischen Originaltextes, den Terry Dawson (terry@perf.no.itg.telecom.com.au) mit viel Sorgfalt zusammengestellt hat. Basis der Übersetzung ist die Version 3.5 vom 16 Januar 1996.

Inhalt

1 Einleitung

Dies ist das *Linux NET-2/3-HOWTO*. Der Text ist eine völlige Neufassung des älteren NET-FAQ und der darauf basierenden *NET-2-HOWTOS* der 1.0+ Versionen. Das Dokument beschreibt den neuen NET-2/3 TCP/IP Netzwerk Code der Linux Kernels von Version 1.0 ab aufwärts.

Es mag seltsam erscheinen, einige Monate nach Erscheinen des 2.0 Kernels noch einen Text zu veröffentlichen, der noch Kernelversionen 1.2.13 und 1.3.x beschreibt. Diese Abschnitte des *Linux NET-2/3-HOWTO* müssen neu geschrieben werden (Terry sitzt gerade daran). Sobald seine neue Version erschienen ist, werde ich mich an die Übersetzung machen... Der Großteil des Textes hat aber auch für die neuen Kernels Gültigkeit.

1.1 Eine Kurze Geschichte des Linux Netzwerk Systems

Ross Biro (`biro@yggdrasil.com`) schrieb den ursprünglichen Netzwerk-Code im Linux Kernel. Er benutzte dazu Ethernet-Treiber von Donald Becker (`becker@cesdis1.gsfc.nasa.gov`), einen SLIP-Treiber von Laurence Culhane (`loz@holmes.demon.co.uk`) und einen D-Link Treiber von Bj0rn Ekwall (`bj0rn@blox.se`).

Die Weiterentwicklung des Linux Netzwerk Codes wurde später von Fred van Kempen (`waltje@hacktic.nl`) übernommen, der basierend auf den Quellen von Ross die NET-2 Version zusammenstellte. NET-2 machte einige Revisionen durch bis zur Version NET-2d. Zu diesem Zeitpunkt begann Alan Cox (`iialan@iifeak.swan.ac.uk`), den Code von Fred nach Fehlern abzusuchen mit dem Ziel, eine wirklich stabiles Code-Paket für die Übernahme in den Standard-Kernel zu schaffen. Dieser Code wurde ursprünglich als NET-2D(ebugged) bezeichnet und wurde kurz vor der Linux Version 1.0 in die Standard-Kernels aufgenommen.

Unterstützung für das PPP-Protokoll wurde von Michael Callahan (`callahan@maths.ox.ac.uk`) und Al Longyear (`longyear@netcom.com`) implementiert, zunächst als ein Satz von Patches, später als fester Bestandteil der Standard-Kernel-Distribution.

Die letzte Revision des Codes, NET-3, erscheint in den Kernelversionen ab 1.1.5. Obwohl prinzipiell noch derselbe Code, wurden viele Fehler behoben und Erweiterungen eingebracht.

Alan hat die IPX und AX.25 Module mitimplementiert, eine auf den neuesten Stand gebrachte Distribution der Netzwerk-Applikationen wurde von Florian La Roche (`flla@stud.uni-sb.de`) zusammengestellt.

NIIBE Yutaka hat den PLIP-Treiber verbessert.

Jonathan Naylor übernahm die Weiterentwicklung des AX.25-Codes und hat viele Dinge wie z.B. die NetROM-Unterstützung hinzugefügt.

Viele weiter Personen haben durch Fehlermeldungen und -Korrekturen sowie durch das Schreiben von Hardware-Treibern mitgeholfen.

2 Disclaimer

Der Linux Netzwerk Code ist eine völlig neue Implementation des TCP/IP-Netzwerkes auf Kernelebene. Er wurde von Grund auf neu geschrieben und ist keine Portierung eines bestehenden Netzwerk-Codes.

Das Netzwerk-System unter Linux ist einer der neuesten und am meisten innovativen Codes, die derzeit bekannt sind. Eine große Zahl an Entwicklern arbeiten daran, ihn für viele neue Anwendungen anzupassen, und aus diesem Grund ist er einem schnellen Wachstum unterworfen. Er unterliegt starken Veränderungen und mag immer noch einige Fehler oder Ungereimtheiten aufweisen, und es werden wohl auch noch Patches zur Fehlerbehebung auftauchen. Wer vor solchen Problemen sicher sein will, sollte bei einer der Versionen bleiben, die in den stabilen Standard-Kernels implementiert sind. Derartige Kernel-Versionen haben eine *gerade* Versionsnummer an der zweiten Stelle, 1.2.7 ist zum Beispiel eine solche, 2.0.0 ebenfalls. Kernel-Versionen mit einer ungeraden Nummer an zweiter Stelle sind **alpha** Versionen und man muß immer damit rechnen, daß ein solcher Kernel Fehler enthält oder einige Probleme bereitet. Es sind eben *Testversionen*. Der eigentliche Netzwerk-Code wird von einer kleinen Gruppe von Leuten entwickelt, einige tausend weitere testen diesen Code, suchen Fehler und liefern auch Vorschläge zu deren Behebung. Wer also selber Probleme hat: Diese sind vermutlich bereits gemeldet und werden bald behoben. Etwas Geduld hilft meistens. Wer auf der anderen Seite in der Lage ist, selber aktiv mitzuarbeiten, sollte seine Mithilfe anbieten.

Ein einzelner kann kaum alles über die Linux Netzwerk Software wissen. Aus diesem Grund ist es möglich, das dieser Text Fehler enthält. Bitte lesen Sie alle README-Dateien, die mit neu zu installierender Software mitgeliefert werden. Diese enthalten meist weitergehende und detailliertere Informationen. Die angegebenen Software-Versionen waren zum Zeitpunkt, als dieser Text geschrieben wurde, die jeweils aktuellsten.

Achtung Obwohl der Name ähnlich wie derjenige der *Berkeley Software Distribution NET-2 release* klingt, hat der Linux Netzwerk Code überhaupt nichts mit diesem zu tun und sollte nicht verwechselt werden.

3 Gibt es bereits Fragen ?

'Die einzige wirklich dämliche Frage ist die, die keiner stellt.'

Wer allgemeine Fragen zur Konfiguration hat, und die Antworten darauf nicht in den diversen *HOWTO* und *FAQ* Texten gefunden hat, ist am besten bedient, wenn er seine Fragen in der Newsgroup `de.comp.os.linux.networking` postet, oder, falls er glaubt, daß es sich um ein spezielles Problem des Linux-Codes handelt, auch in der `NET` mailing list (in Englisch). Auf jeden Fall: **Geben Sie so viele Informationen wie möglich an**, denn es ist nicht ärgerlicher als ein Fehlerbericht, bei dem man nicht die leiseste Idee hat, wo man mit der Suche beginnen soll!

Die Versionsnummern und Revisionen des verwendeten Codes sowie die genaue Beschreibung des Fehlers und der Umstände, unter denen er auftritt, sind wesentlich. Ergebnisse eines trace-Aufrufs sowie die Ausgabe von debug-Meldungen sollten wo immer möglich ebenfalls angegeben werden.

Wer Fragen zur Konfiguration irgendeiner Linux-Distribution hat, oder Probleme mit deren Netzwerk-Teil hat, sollte auf jeden Fall zuerst die Autoren/Vertreiber dieser Distribution kontaktieren, bevor er das Problem an die Entwickler des Netzwerk-Codes schickt. Der Grund hierfür ist, daß einige Distributionen eine nicht dem Standard entsprechende Verzeichnis-Struktur aufweisen oder nicht-standardisierte oder sogar Testversionen des Codes und der Hilfsprogramme verwenden. Die Entwickler des NET-2 Codes können nur für den Code, wie er hier beschrieben ist, Unterstützung geben, bzw. entsprechend dem in den Instruktionen für Alpha-Tester angegebenen Rahmen.

Zur Aufnahme in die Linux Mailing-Liste `linux-net` genügt eine email an:

```
Majordomo@vger.rutgers.edu
```

mit der Zeile:

```
subscribe linux-net
```

als Text (die `subject:`-Zeile wird ignoriert). Aber denken Sie dran: `linux-net` ist ausschließlich zur Diskussionen rund um die Software-Entwicklung gedacht!

Weiterhin gibt es inzwischen eine PPP-Liste. Um daran teilzunehmen gelte dieselben Regeln wie gerade bei `linux-net` angegeben, lediglich muß im Text der email stattdessen `subscribe linux-ppp` angegeben werden.

In einer weiteren Liste, `linux-hams`, werden Programme und Probleme diskutiert, die sich mit Amateur Radio befassen. Zur Teilnahme gilt wiederum: email an die obengenannte Adresse mit dem Text `subscribe linux-hams` .

4 Weiterführende Literatur (über TCP/IP)

Wer nach Informationen über das TCP/IP-Netzwerk-Protokoll sucht, die in diesem *HOWTO* nicht behandelt werden, wird vielleicht an den folgenden Stellen fündig. Sie enthalten einige sehr nützliche Informationen..

Olaf Kirch hat im Rahmen des *Linux Documentation Project* ein substantielles Dokument mit dem Namen *Linux Network Administration Guide* verfaßt. Es ist ein exzellentes Buch, welches alle Aspekte des TCP/IP unter Linux abdeckt, inklusive NFS, UUCP, mail, News, nameserver usw.

Olaf's Buch ist in gewisser Weise eine Ergänzung zu diesem *HOWTO*, da es dort einsetzt, wo dieser Text aufhört. Es behandelt die Installation und Konfiguration des Netzwerk-Codes, also 'wie man seinen Rechner an's Netz hängt'. Wer sich noch nicht mit dem Unix-Netzwerk auskennt, dem kann nur wärmstens empfohlen werden, sich eine Kopie davon zu besorgen. Es wird sicherlich einige der Fragen beantworten, deren Beantwortung den Rahmen dieses *HOWTO* übersteigen würde.

Die aktuelle Version ist via ftp erhältlich:

```
sunsite.unc.edu:/pub/Linux/docs/linux-doc-project/network-guide/
```

In diesem Verzeichnis befinden sich unterschiedliche Versionen des *NAG*. Die gängigen Formate werden unterstützt: reines ASCII, PostScript, DVI, LaTeX und groff.

Der *Linux Network Administrators Guide* ist Copyright (c) by Olaf Kirch.

Es gibt inzwischen einige Verlage, die die Linux Dokumentationen vertreiben. Wem also das Herunterladen über ftp und ausdrucken auf dem Drucker zu viel Aufwand ist, sollte Olaf's Buch in gebundener Form in jedem besseren Buchladen finden. Inzwischen ist es auch in deutscher Sprache erhältlich.

Außerdem sollte man die anderen für Netzwerke relevanten Linux *HOWTO*s lesen.

Dies sind:

Das *Ethernet-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO.html`), das jeder lesen sollte, der eine Ethernet-Karte unter Linux benutzen will. Es liefert jede Menge Hilfen bei der Auswahl, Installation und Konfiguration einer Netzwerkkarte unter Linux und Anleitungen, wie man Probleme mit den Treibern diagnostizieren kann.

Das *PPP-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/PPP-HOWTO.html`) für alle an PPP Interessierten.

Das *IPX-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/IPX-HOWTO.html`) gibt Informationen über die IPX-Unterstützung in Linux.

Das *Serial-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Serial-HOWTO.html`), falls SLIP oder PPP im Server-Modus betrieben werden sollen.

Das *NIS-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/NIS-HOWTO.html`). Information über den Betrieb einer Version des Network Information Service von Sun.

Das *HAM-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/HAM-HOWTO.html`) für diejenigen, die die Amateur Radio Software unter Linux benutzen wollen.

Das *Mail-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Mail-HOWTO.html`) und das *News-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/News-HOWTO.html`) geben genauere Anleitungen, wie Mail und News korrekt aufgesetzt werden.

Das *UUCP-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/UUCP-HOWTO.html`), falls die Verbindung zum Internet über UUCP erfolgt.

Das *Firewall-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Firewall-HOWTO.html`) für alle, die ihr Netzwerk durch einen Linux-Rechner als Firewall nach außen absichern wollen.

Wer einige Informationen zum TCP/IP-Netzwerk generell sucht, dem können die folgenden Dokumente empfohlen werden:

tcp/ip introduction

Text Version (`ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc`),

PostScript Version (`ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps`).

tcp/ip administration

Text Version (`ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc`),

PostScript Version (`ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps`).

Eine sehr viel detaillierteres Buch zum Thema TCP/IP ist:

```
"Internetworking with TCP/IP"
by Douglas E. Comer

ISBN 0-13-474321-0
Prentice Hall publications.
```

Wer selber Netzwerk-Anwendungen unter Unix schreiben will, sollte folgendes Buch unbedingt lesen:

```
"Unix Network Programming"
by W. Richard Stevens

ISBN 0-13-949876-1
Prentice Hall publications.
```

4.1 Aktuelle Versionen dieses Dokuments

Die Netzwerk-Unterstützung für Linux ändert sich sehr schnell, mit laufend neuen Erweiterungen und Verbesserungen. Gleiches muß deshalb auch für diesen Text gelten. Ist er bereits deutlich älter als ein halbes Jahr ist mittlerweile vielleicht bereits eine aktuellere Version erschienen, nachsehen lohnt sich in diesem Fall!

Die neueste englische Version des *NET-2-HOWTO* kann immer über anonymous ftp bezogen werden:

```
sunsite.unc.edu:/pub/Linux/docs/HOWTO/NET-2-HOWTO
```

Die jeweils aktuelle deutsche Version gibt es hier:

`ftp.heise.de:/pub`

oder

`ftp.uni-stuttgart.de:/pub/systems/linux/local/doc/`

oder via World Wide Web auf der Homepage des *Deutschen Linux HOWTO Projekts* (`http://www.tu-harburg.de/~semb2204/dlhp/index.html`).

Außerdem werden beide Versionen des *NET-2-HOWTO* regelmäßig in den folgenden Newsgroups gepostet: `comp.os.linux.networking`, `comp.os.linux.answers` und `news.answers`.

Die FAQ-Postings aus `news.answers`, also auch dieses *HOWTO*, werden auf `rtfm.mit.edu:/pub/usenet` archiviert.

4.2 Feedback

Korrekturen und Vorschläge zu dieser deutschen Version bitte an mich (`pit@uni-sw.gwdg.de`) senden. Ich werde dann Dinge, die nicht auf meine Kappe gehen, an den Autor des englischen Originals weiterleiten.

5 Einige in diesem Dokument verwendete Begriffe

In diesem Text werden sehr oft die Begriffe *Klient* und *Server* verwendet. Das sind normalerweise recht genau definierte Begriffe, die aber in diesem Text sehr viel allgemeiner benutzt werden. Sie sollen im Folgenden bedeuten:

Klient

Derjenige Rechner oder das Programm, das eine Aktion oder Verbindung initiiert, um einen Service in Anspruch zu nehmen oder Daten zu empfangen.

Server

Derjenige Rechner oder das Programm, welches die eingehenden Verbindungen von verschiedenen Rechnern im Netzwerk entgegennimmt und die verlangten Services oder Daten zur Verfügung stellt.

Diese Definitionen sind zwar auch nicht sehr genau, aber immerhin erlauben sie eine eindeutige Unterscheidung der beiden Enden eines Peer-to-Peer-Systems wie zum Beispiel SLIP oder PPP, bei denen es in Wirklichkeit so etwas wie Klient und Server nicht gibt.

Weiter Begriffe, die auftauchen werden, sind:

IP-Adresse

Dies ist eine Nummer, die einen TCP/IP Rechner in einem Netzwerk eindeutig identifiziert. Diese Adresse ist 4 Bytes lang und wird normalerweise in einer punktierten Dezimalschreibweise genannt, bei der der Wert eines jeden Bytes, getrennt durch Dezimalpunkte '`.`', angegeben wird.

Hardware-Adresse

Dies ist eine Nummer, die den jeweiligen Rechner in einem Netzwerk auf dem untersten Zugriffs-Niveau eindeutig identifiziert. Beispiele hierfür sind die *Ethernet-Adresse* oder die *AX.25-Adresse*.

Datagramm

Ein Datagramm ist ein einzelnes Datenpaket, bestehend aus den eigentlichen Daten sowie einem Header, der die Adresse enthält. Sie sind die Basiseinheiten der Verbindungen über ein IP-Netzwerk. Manchmal wird ein Datagramm auch als Paket bezeichnet.

MTU

Die maximale Übertragungseinheit (**M**aximum **T**ransmission **U**nit) ist ein Parameter der festlegt, wie groß ein Datagramm maximal sein darf, das über eine IP-Schnittstelle weitergegeben werden kann, ohne daß es in kleinere Pakete zerlegt werden muß. Die MTU sollte größer sein als das größte Datagramm, welches man unfragmentiert senden möchte. Dies verhindert eine solche Fragmentierung allerdings nur lokal, da ein anderer Rechner in der Übertragungskette eine kleinere MTU haben kann, sodaß das Paket dann dort aufgespalten wird. Typische Werte für die MTU sind 1500 für eine Ethernet-Schnittstelle oder 576 für SLIP.

MSS

Die maximale Segmentgröße (**M**aximum **S**egment **S**ize) ist die größte Datenmenge, die auf einmal übertragen werden kann. Um lokale Fragmentierung zu vermeiden sollte MSS die Differenz aus MTU und IP-Header-Größe sein.

Window

Das *Window* (Fenster) ist die größte Datenmenge, die ein Empfänger zum jeweiligen Zeitpunkt entgegennehmen kann.

Route

Die *Route* ist der Weg, den die Datagramme auf ihrem Weg zur Empfangsstation nehmen.

ARP

Dies ist ein Akronym für das *Address Resolution Protokoll* welches festlegt, wie ein Netzwerkrechner eine IP-Adresse in eine Hardware-Adresse übersetzt.

6 Von NET-3 unterstützte Funktionalität

Der NET Code ist eine vollständige, Kernel-basierte Implementation von TCP/IP für Linux.

6.1 Generelle Unterstützung

Ethernet-Karten

Die meisten gängigen Ethernet-Karten werden unterstützt, inklusive einiger tragbarer, Pocket-Adapter und PCI-Karten.

SLIP (Serial Line IP) und PPP (Point to Point Protocol)

zur Realisierung eines TCP/IP-Netzes über serielle Verbindungen wie Modems oder lokale Kabelverbindungen.

Van Jacobsen Header Komprimierung

packt und komprimiert die TCP/IP Header Blöcke, um die Leistungsfähigkeit von SLIP/PPP zu erhöhen.

PLIP (Parallel Line IP)

zur Verbindung zweier Rechner über den parallelen Druckerport.

EQL Load Balancing

Erlaubt es, zwei oder mehr Verbindungen simultan zwischen zwei Maschinen aufzubauen und so den Datendurchsatz effektiv zu vervielfachen. Dies wird nur von neueren Kernel-Versionen unterstützt.

NFS (Networked File System)

erlaubt das mounten von Partitionen entfernter Rechner über eine Netzwerkverbindung.

IPX (Novell)

erlaubt es, IPX Anwendungen zu schreiben und Linux als Router in einem IPX-Netzwerk zu betreiben.

NIS (Sun's Network Information System)

eine Implementation des NIS ist auch für Linux verfügbar.

ARCNet

neuere Kernels unterstützen auch ARCNet. Es ist zwar langsamer als Ethernet, dafür ist die Hardware (Karten) billiger.

IBM's Token Ring

ein experimenteller Token-Ring-Treiber ist Bestandteil der aktuellen Kernels, damit kann ein Linux-Rechner in einem entsprechenden lokalen Netzwerk (LAN) betrieben werden.

AppleTalk

(oder heißt es EtherTalk?) Egal, ich glaube es ermöglicht es, Dateien und Drucker gemeinsam mit einem Macintosh zu verwenden. Nähere Hinweise gibt es weiter unten im Abschnitt 'Experimentelle und in der Entwicklung befindliche Module'.

WaveLan drahtloses Netzwerk

die 'WaveLan wireless Lan Card' wird von den aktuellen Kernelversionen unterstützt. Sie erlaubt es, einen Linux-Rechner mobil und in einiger Entfernung zum Netzwerk zu betreiben.

ISDN

es gibt experimentelle Unterstützung für einige ISDN-Karten. Nähere Hinweise gibt es weiter unten im Abschnitt 'Experimentelle und in der Entwicklung befindliche Module'.

ATM

eine Gruppe von Programmierern arbeitet an der Unterstützung von ATM.

IP Firewall

hilft, den Linux-Rechner für ein Netzwerk als sicheren Gateway nach außen zu konfigurieren.

IP Accounting

erlaubt die genaue Aufschlüsselung, wer welche Netzwerk-Dienste in Anspruch nimmt.

IP Tunneling

interessant um mit IP zu experimentieren.

Gegenwärtig wird vom NET-3-Code *nicht* unterstützt:

SPX/NCP (Novell Netware)

würde es Linux erlauben, Novell Netzwerk Dateisysteme als Server oder Klient zu benutzen und auf Druckern im Novell Netz zu drucken. An diesem Projekt wird zwar gearbeitet, aber durch den propietären Charakter des Novell Netzwerkes kann das noch einige Zeit brauchen.

FDDI

FDDI-Karten werden meines Wissens von Linux derzeit nicht unterstützt.

System V Streams

einige Leute arbeiten an Sys-V Streams für Linux, Details wurden aber bislang noch nicht publik gemacht.

6.2 Unterstützte Ethernet-Karten

Der 1.2.0 Kernel Release unterstützt die folgenden Karten:

- WD80*3 und dazu Kompatible.
- SMC Ultra.
- AMD LANCE und PCnet (AT1500 and NE2100) und dazu Kompatible.
- 3Com 3c501 (veraltet und sehr langsam).
- 3Com 3c503.
- 3Com 3c505.
- 3Com 3c507.
- 3Com 3c509/3c579.
- Cabletron E21xx.
- DEPCA und dazu Kompatible.
- EtherWorks 3.
- ARCNet.
- AT1700 (keine Nachbauten).
- EtherExpress.
- NI5210 und dazu Kompatible.
- NI6510.
- WaveLAN.
- HP PCLAN+ (27247B and 27252A).
- HP PCLAN (27245 und weitere Modelle der 27xxx Serien).
- NE2000/NE1000 und dazu Kompatible
- SK_G16.
- Ansel Communications EISA 3200.
- Apricot Xen-II on board ethernet.
- DE425, DE434, DE435.
- Zenith Z-Note.
- AT-LAN-TEC/RealTek Pocket-Adapter.
- D-Link DE600 Pocket-Adapter und dazu kompatible.
- D-Link DE620 Pocket-Adapter und dazu kompatible.

Spätere Kernelversionen unterstützen vermutlich eine noch größere Menge an Karten.

Wer beabsichtigt, unter Linux eine Ethernet-Karte zu betreiben, sollte auch das *Ethernet-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO.html`) lesen, da es viele nützliche Informationen über unterstützte Karten enthält, vor allem auch für diejenigen, die eine Ethernet-Karte speziell für Linux kaufen wollen.

Wie bereits erwähnt unterstützt Linux auch einige andere Netzwerk-Protokolle, falls kein Zugang zu einem Ethernet oder einer entsprechenden Netzwerkkarte besteht. Viele Universitäten und Firmen bieten weltweit die eine oder andere Möglichkeit zu einem Dial-Up-Zugang zum Internet an. Normalerweise wird dabei ein Zugang entweder via SLIP oder PPP zur Verfügung gestellt, sodaß man kaum eine andere Wahl als diese Protokolle besitzt. Alles was man dazu benötigt ist ein Telefonanschluß, ein Modem (ein bereits vorhandenes mag für den Anfang sicherlich gut genug sein) und ein entsprechend konfiguriertes Linux-System. Im weiteren Verlauf dieses Textes beschäftigen sich einige Abschnitte damit und zeigen genau auf, was man dazu braucht.

6.3 Unterstützung für Amateur Radio

Inzwischen unterstützt Linux einige Dinge die speziell für Amateur Radio gedacht sind. Die neuesten Kernels schließen Standardmäßig ein:

AX.25

Alan Cox und Jonathan Naylor haben funktionierende AX.25-Sockets im Kernel implementiert.

NetRom

Jonathan Naylor hat eine kernelbasierte NetRom-Unterstützung implementiert. Sie ist noch im Versuchsstadium, macht aber gute Fortschritte.

Ottawa PI Karte

Dave Perry von der Ottawa Packet Radio Group hat einen funktionierenden Treiber für diese Karte geschrieben.

SCC Karten

ein generischer Treiber für SCC-Karten ist Bestandteil der neuesten Kernels. Er wird von Jörg Reuter entwickelt.

Ein Kernel-Nutzer-Verbindungs Treiber

er erlaubt die Kommunikation von Nutzerprogrammen mit dem Kernel ohne störende pty's.

Weitere Details zur Amateur Radio Unterstützung finden sich im *HAM-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/HAM-HOWTO.html`).

7 Woher bekomme ich die NET-2/NET-3 Software ?

Bevor mit der eigentlichen Konfigurationsarbeit begonnen werden kann muß sichergestellt werden, daß das komplette Netzwerk-Paket vorhanden ist. Das umfaßt eine aktuelle Kernelversion (1.0 oder neuer), die richtigen Libraries, die TCP/IP Konfigurationsprogramme und -Dateien wie z.B. `/sbin/ifconfig` und `/etc/hosts`, und schließlich auch eine Reihe von Anwendungsprogrammen wie `telnet`, `ftp`, `rlogin` usw.

Wer ein komplettes Linux-System über einen Distributor bezogen hat, besitzt höchstwahrscheinlich schon alles, was er benötigt. Viele Linux-Distributionen sind von vorneherein mit den notwendigen Konfigurationsdateien, ausführbaren Programmen, Bibliotheken und Kernelversionen ausgestattet. Dann erübrigt es sich, sich die folgenden Dateien zu besorgen.

ACHTUNG: Es kann durchaus sein, daß sich manche Dateien an einer anderen als der in diesem *HOWTO* angegebenen Stelle im Verzeichnisbaum befinden.

Wer bereits die vollständige Netzwerk-Software besitzt, kann jetzt direkt zum Abschnitt Kernelkonfiguration weitergehen. Alle anderen sollten die folgenden Anweisungen befolgen.

7.1 Der Kernel Quellcode

Die Versionen 1.2.x des Linux-Kernels sind die **stabilen** Versionen, sie werden auch gerne als 'Production Release' bezeichnet. Die Kernels 1.3.*, und allgemein alle Kernels mit ungerader zweiter Ziffer, sind **Entwicklerversionen**. Wer keine Lust hat, sich mit dem Einspielen von Patches und Neukompilieren von Kernels zu befassen, sollte bei den stabilen Versionen bleiben. Dennoch, gerade im Falle des Netzwerk-Codes kann ich nur jedem empfehlen, trotzdem den geringen Aufwand in Kauf zu nehmen und einen der aktuellen Releases zu verwenden, da gerade hier viele Veränderungen und vor allem Verbesserungen am Netzwerk-Code vorgenommen wurden. Es wird zwar immer und überall erwähnt, trotzdem: Wer einen neuen Kernel ausprobieren will, sollte eine aktuelle Sicherungskopie seines Systems besitzen, nur für den Fall das dabei etwas Schwerwiegendes daneben geht.

Die aktuelle Kernelversion kann gefunden werden auf

```
ftp.funet.fi:/pub/OS/Linux/PEOPLE/Linus/v1.2/linux-1.2.13.tar.gz
```

Es handelt sich um ein gezipptes TAR-Archiv, man benötigt deshalb das Programm *gzip* zum Wiederauspacken.

Zum Auspacken der Kernel-Quellcodes sollte man so vorgehen:

```
% cd /usr/src
% mv linux linux.old
% gzip -dc linux-1.2.13.tar.gz | tar xvf -
```

Im selben Verzeichnis befinden sich auch einige Dateien `patch-1.2.1.gz ...`. Es handelt sich dabei um Patch-Dateien. Eine Kernelversion 1.2.1 besagt z.B. daß es sich um die Version 1.2.0 mit dem eingespielten Patch 1.2.1 handelt. Wenn sich also im Verzeichnis Patch-Dateien befinden, deren Versionsnummer größer ist als die des gegenwärtig installierten Kernels, so müssen **alle** diese Patch-Dateien heruntergeladen und **hintereinander** eingespielt werden. Dies geschieht jeweils mit dem Befehl

```
% cd /usr/src
% for patchfile in .../patch*
> do
> gzip -dc $patchfile | patch -p0 2>>patch.errs
> done

  ...
```

Die dabei erzeugte Datei `patch.errs` muß dann nach dem Wort `fail` durchsucht werden. Taucht es nicht in dieser Datei auf, ist der Patch einwandfrei abgelaufen. Andernfalls sollte man sich eine Originale Kernelversion besorgen und die Patches einen nach dem anderen einspielen, bis derjenige Patch gefunden ist, der den Fehler verursacht hat. Erst wenn man nicht herausfinden kann, was schief gegangen ist, kann man einen Fehlerbericht schreiben.

7.2 Die Bibliotheken (libraries)

Es empfiehlt sich, **mindestens** die Version 4.4.2 der `libc` zu verwenden, da frühere Versionen einige Probleme mit Subnet-Masken hatten.

Die aktuelle a.out-Version (libc-4.6.27) befindet sich auf:

```
sunsite.unc.edu:/pub/Linux/GCC/
```

Außerdem benötigt man mindestens folgende Dateien:

- image-4.6.27.tar.gz
- inc-4.6.27.tar.gz
- extra-4.6.27.tar.gz
- release.libc-4.6.27

Vor der Installation der neuen libc-Version **muß** man unbedingt die Release-Notes **release.libc-4.6.27** lesen. Außerdem sollten für libc-Versionen ab 4.6.26 mindestens GCC Version 2.6.2 und Kernelversionen ab 1.1.52 verwendet werden.

7.3 Das Programmpaket zur Netzwerk-Konfiguration

Auf jeden Fall wird das Paket der Hilfsprogramme zur Netzwerk-Konfigurierung benötigt. Die aktuelle Version bekommt man hier:

```
ftp.linux.org.uk:/pub/linux/Networking/PROGRAMS/NetTools/net-tools-1.2.0.tar.gz
```

Da sich der Netzwerk-Code im Kernel immer noch recht häufig ändert sind auch für diese Hilfsprogramme Anpassungen notwendig. Man muß sich deshalb die für den benutzten Kernel ausgewiesenen Pakete besorgen.

Die Dateinamen geben dabei die erste Versionsnummer des Kernels an, ab der die Programme funktionieren. Die richtige Datei ist also diejenige mit derselben oder der nächstkleineren Versionsnummer wie der benutzte Kernel.

Um die Hilfsprogramme zu kompilieren und installieren sollte man folgendes machen:

```
% cd /usr/src
% mkdir net-tools
% cd net-tools
% gzip -dc net-tools-1.2.0.tar.gz | tar xvf -
% make
```

Dadurch wird automatisch das Shell-Script `Configure.sh` aufgerufen. Wenn der make-Prozeß erfolgreich beendet wird, beendet

```
% make install
```

die Installation.

Wer eine Kernel-Version vor 1.1.26 benutzt, sollte sich auch folgendes ansehen:

```
ftp.linux.org.uk:/pub/linux/Networking/PROGRAMS/Other/net032
```

In diesem Verzeichnis befinden sich drei Versionen der Netzwerk-Programme. Die folgende Tabelle zeigt, für welche Kernel-Versionen die jeweiligen Pakete gedacht sind:

net-0.32d-net3.tar.gz	1.1.12+
net-0.32b.tar.gz	1.1.4+
net-0.32.old.tar.gz	vor 1.1.4

Die Pakete enthalten die wichtigsten Konfigurationsprogramme wie `ifconfig`, `route`, `netstat` usw. Deren Benutzung wird später erläutert.

7.4 Anwendungsprogramme

Selbstverständlich benötigt man auch eine Reihe von Applikationen, die die Netzwerkfähigkeiten auch ausnutzen, wie z.B. `telnet`, `ftp` oder `finger` und ihre entsprechenden Dämonen. Florian La Roche, (`flla@stud.uni-sb.de`) hat eine sehr umfassende Distribution dieser Programme zusammengestellt, die sowohl in Binärform wie auch als Quellcode erhältlich ist. Die kompilierten TCP/IP-Applikationen und einige beispielhafte Konfigurationsdateien befinden sich in:

ftp.funet.fi

```
/pub/OS/Linux/PEOPLE/Linus/net-source/base/NetKit-A-0.08.bin.tar.gz
/pub/OS/Linux/PEOPLE/Linus/net-source/base/NetKit-B-0.06.tar.gz
```

Es sollten jeweils die neuesten Versionen verwendet werden. Aber unbedingt zunächst die Datei `README` lesen um sicherzustellen, daß das eigene System auch die notwendigen Voraussetzungen für die neue Version erfüllt.

Die von Florian zusammengestellte Binär-Distribution (B-Datei) gibt es derzeit nicht mehr, deshalb müssen die Quellen selber kompiliert werden. Dies geschieht so:

```
% cd /usr/src
% gzip -dc NetKit-B-0.06.tar.gz | tar xpvlf -
% cd NetKit-B-0.06
```

Dann muß das `Makefile` editiert und angepaßt werden. Die Datei `README` gibt hierzu Hinweise. Wichtig ist, daß die Variable `HAVE_SHADOW_PASSWORDS` richtig definiert wird. Wer keine benutzt, sollte den entsprechenden Eintrag im Makefile durch ein # am Zeilenanfang auskommentieren. Der Rest sollte nicht verändert werden müssen, deshalb genügt nun ein

```
% make
% make install
```

um die Installation abzuschließen.

Wichtiger Hinweis: Florian hat diese TAR-Archive zusammengestellt und zur leichteren Installation mit den `net-tools-n.n.nn` auch ein vorkompiliertes Paket erstellt. Leider hat er aber hierfür eine andere Verzeichnis-Struktur zugrundegelegt als dies Alan bei der Zusammenstellung der Installations-Scripte der net-tools tat. Dies bedeutet, daß man bei der Installation besondere Vorsicht walten lassen sollte. Dies wird von Florian demnächst angepaßt, so daß es auf Dauer kein Problem sein wird. Aber bis dahin empfehle ich statt der oben angegebenen Prozedur die folgende:

```
-  Auspacken der Programme an einem sicheren Ort
% cd /usr/src
% mkdir NetKit
% cd NetKit
% gzip -dc NetKit-A-0.07.bin.tar.gz | tar xpvlf -
% gzip -dc NetKit-B-0.06.bin.tar.gz | tar xpvlf -

-  Loeschen von Florians Versionen der erwaehnten Konfigurationsprogramme
% rm ./bin/hostname ./sbin/route ./sbin/ifconfig ./sbin/netstat
% rm ./usr/sbin/arp ./usr/sbin/rarp ./usr/sbin/slattach

-  Kopieren der Programme in ihre endgueltigen Verzeichnisse
% cp -vrpd . /
```

7.5 Zusätzliche Treiber oder Pakete

Wer zusätzliche, im Alpha oder Beta Stadium befindlichen Code oder Treiber verwenden will oder muß, der noch nicht in der Standard-Distribution des Kernels mitaufgenommen wurde, muß die entsprechende Software separat besorgen und installieren. In den entsprechenden Abschnitten dieses *HOWTO* sind dazu genauere Angaben gemacht.

8 Konfiguration des Kernels

Bevor mit der Arbeit der Konfiguration des Netzwerkes begonnen werden kann muß sichergestellt sein, daß der verwendete Kernel die gewünschten Netzwerkprotokolle und Hardware unterstützt. Der sicherste Weg hierfür besteht darin, sich seinen Kernel selber zu kompilieren. Dies ist generell anzuraten, da auf diese Weise nicht benötigte Treiber weggelassen werden können und der resultierende Kernel weniger Speicher belegt.

Vorausgesetzt, die Kernel-Quellen sind bereits an der richtigen Stelle entpackt und die benötigten Patches eingespielt worden, sollte zunächst ein Blick in die Datei `/usr/src/linux/drivers/net/CONFIG` geworfen werden. Sie enthält viele Kommentare, die bei der Auswahl der Optionen helfen, aber normalerweise sind keine Veränderungen notwendig. Wirklich notwendig sind Änderungen nur bei exotischen Ethernet-Karten oder bei solchen, die von den Treibern nicht automatisch erkannt werden. In diesem Fall können einige Einstellungen der Karte direkt im Code festgelegt werden. Ein Beispiel sind die (Fast-)nachbauten der WD-8013, hier muß die Adresse des Shared-Memory-Bereiches eingestellt werden, damit auch diese Karten korrekt erkannt und betrieben werden können. Ebenfalls in dieser Datei können einige Parameter für den PLIP-Treiber eingestellt werden, jedoch sind die Standard-Einstellungen außer für extrem langsame Rechner ausreichend.

Das *Ethernet-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO`) gibt genauere Informationen über diese Datei und darüber, wie sie das Verhalten der Treiber beeinflußt.

Ist die Konfigurationsdatei zufriedenstellend zusammengestellt, kann mit der eigentlichen Kernel-Kompilation begonnen werden. Der erste Schritt besteht darin, das Makefile im obersten Verzeichnis zu editieren um Sicherzustellen, daß die korrekten VAG-Einstellungen verwendet werden. Danach muß man das Kernel-Konfigurationsprogramm starten:

```
% cd /usr/src/linux
% make config
```

Nun werden eine ganze Reihe an Fragen gestellt. Sie gliedern sich in vier Hauptgruppen: *Generelle Einstellungen*, *Netzwerk-Einstellungen*, *Netzwerk-Hardware-Unterstützung* und *Dateisysteme*. Der komplexeste ist der Abschnitt ist die Hardware-Unterstützung für das Netzwerk, da hier die richtigen Geräte ausgewählt werden müssen, die hinterher unterstützt werden. Für die meisten anderen Einstellungen können die vorgegebenen Werte ohne große Bedenken verwendet werden. Der Ablauf wird in etwa so aussehen:

```
*
* General setup
*
 ...
 ...
Networking support (CONFIG_NET) [y] y
 ...
 ...
```

Im allgemeinen Abschnitt wird zunächst nur festgelegt, ob überhaupt Netzwerk-Unterstützung gewünscht ist. Selbstverständlich muß man darauf mit Ja antworten ;-)

```
*
* Networking options
```

```
*
TCP/IP networking (CONFIG_INET) [y]
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n]
IP multicasting (CONFIG_IP_MULTICAST) [n]
IP firewalling (CONFIG_IP_FIREWALL) [n]
IP accounting (CONFIG_IP_ACCT) [n]
*
* (it is safe to leave these untouched)
*
PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n]
Reverse ARP (CONFIG_INET_RARP) [n]
Assume subnets are local (CONFIG_INET_SNARL) [y]
Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n]
The IPX protocol (CONFIG_IPX) [n]
*
```

Die zweite Hälfte des Abschnittes *Netzwerk-Optionen* erlaubt es, einige seltsame Dinge an- oder abzuschalten. Die Standardeinstellungen sind hier immer die sichere Wahl. Was sie im einzelnen bedeuten, wird später noch erklärt. Doch wer andere als die Standard-Einstellung verwenden will sollte schon genau wissen, was er vorhat.

```
*
*
* Network device support
*
Network device support? (CONFIG_NETDEVICES) [y]
Dummy net driver support (CONFIG_DUMMY) [n]
SLIP (serial line) support (CONFIG_SLIP) [y]
 CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [y]
 16 channels instead of 4 (SL_SLIP_LOTS) [n]
PPP (point-to-point) support (CONFIG_PPP) [y]
PLIP (parallel port) support (CONFIG_PLIP) [n]
Do you want to be offered ALPHA test drivers (CONFIG_NET_ALPHA) [n]
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [y]
WD80*3 support (CONFIG_WD80x3) [y]
SMC Ultra support (CONFIG_ULTRA) [n]
AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [n]
3COM cards (CONFIG_NET_VENDOR_3COM) [n]
Other ISA cards (CONFIG_NET_ISA) [n]
EISA, VLB, PCI and on board controllers (CONFIG_NET_EISA) [n]
Pocket and portable adaptors (CONFIG_NET_POCKET) [n]
*
```

Dieser Abschnitt ist der wichtigste. Hier wird eingestellt, welche Hardware-Unterstützung eingebunden werden soll. Im obenstehenden Beispiel sind Unterstützung für SLIP mit komprimierten Heraderdateien, für PPP und der WD80*3-Treiber ausgewählt, und sonst alles andere abgeschaltet. Weitere Optionen können auftauchen, je nachdem, welche Antworten man gibt. So taucht z.B. die Frage nach dem komprimierten SLIP und den 16 Kanälen nicht auf, wenn man bei der Frage für `SLIP...` mit `n` antwortet. Generell sollte man bei all den Dingen, mit denen man herumspielen will, mit `y` antworten und alle anderen mit `n` abschalten.

```
*
* Filesystems
*
 ...
 ...
/proc filesystem support (CONFIG_PROC_FS) [y]
NFS filesystem support (CONFIG_NFS_FS) [y]
```

```
...
...
```

Wer einen NFS-Klienten aufbauen will, um Dateisysteme von anderen Rechnern zu importieren, der muß auf die Frage nach Unterstützung für NFS mit `y` antworten. Meist ist es auch notwendig, das `/proc`-Filesystem einzubinden, da es von einigen Netzwerkprogrammen benutzt wird.

Nach vollendeter Konfiguration verbleibt es nur noch, den Kernel auch wirklich zu kompilieren:

```
% make dep
% make
```

Geht alles glatt, darf man ein abschließendes `make zlilo` nicht vergessen, damit der neue Kernel auch installiert wird.

8.1 Was bedeuten denn alle diese Netzwerk-Optionen ?

Die neueren Kernels haben eine Vielzahl an Konfigurations-Optionen, die bei einem `make config` abgefragt werden. Normalerweise müssen sie Standard-Einstellungen nicht verändert werden, doch einige Optionen können unter Umständen hilfreich sein.

TCP/IP Netzwerk

Dies ist offensichtlich: Damit wird eingestellt, ob der Kernel das TCP/IP-Protokoll unterstützt. Wer dieses HOWTO liest, wird mit ziemlicher Sicherheit mit `y` darauf antworten.

Dummy Networking Device

Dies ist ein Pseudo-Gerät. Es wurde eingeführt, damit SLIP und PPP-Nutzer ihren Rechner auch dann auf eine Netzwerkadresse konfigurieren können, wenn gerade keine serielle Verbindung zum Provider besteht. Es stellt weiterhin eine sehr einfache Möglichkeit dar, einen Linux-Rechner mit zwei verschiedenen Adressen zu versehen.

IP forwarding/gatewaying

Diese Einstellung legt fest, was der Linux-Kernel mit einem Datagramm macht, das an keines der konfigurierten Geräte adressiert ist. Ist sie abgeschaltet, so ignoriert der Kernel solche Pakete. Wer seinen Rechner als Router verwenden will, **muß** diese Option aktivieren. Dies gilt z.B. auch für SLIP und PPP Server.

IP multicasting

Dabei handelt es sich um eine experimentelle Unterstützung von Anwendungen wie 'Internet Talk Radio' oder 'Live Video'. Es ist jedoch eine reine Kernel-Unterstützung für die benötigten Protokolle, um sie zu benutzen sind spezielle Anwendungsprogramme nötig.

IP Firewalling

Diese Einstellung ermöglicht es, sehr flexible Sicherheitsschranken für das Linux-Netzwerk zu konfigurieren. So können z.B. TCP/IP-Verbindungen auf der Basis der Netzwerkadressen erlaubt und abgeschaltet werden. Es sind aber auch zusätzliche Programme nötig, um diese Möglichkeiten zu nutzen.

IP Accounting

Diese Option kann von Leuten benutzt werden, die mit ihrem Linux-Rechner Internet-Verbindungen auf kommerzieller Basis anbieten wollen, z.B. über einen SLIP-Server. Sie erlaubt es, ein- und ausgehende Datagramme nach Adresse oder serieller Verbindung zu zählen. Mit der geeigneten Software können so detaillierte Rechnungen für jeden Nutzer des Netzwerkes aufgestellt werden.

PC/TCP Compatibility Mode

Dieser Modus bietet eine Umgehung von Problemen, die auftauchen, wenn PC/TCP-Netzwerk-Programme mit dem Linux-Rechner kommunizieren wollen: Es gibt einen Fehler im PC/TCP-Protokoll, der wiederum einen schwer zu beseitigenden Fehler auf der Linux-Seite hervorruft. Diese Option verhindert, daß solche Probleme auftreten. Normalerweise wird sie nicht benötigt und bleibt ausgeschaltet. Wer jedoch in seinem Netz Benutzer hat, die PC/TCP verwenden, sollte diesen Modus aktivieren, um Problemen vorzubeugen.

Reverse ARP

Damit kann das Protokoll zur umgekehrten Adress-Auflösung (RARP) in den Kernel aufgenommen werden. Es wurde eingeführt, damit Sun-3 Systeme booten können. Für andere Rechner ist es normalerweise nicht sinnvoll.

Assume subnets are local

Hiermit wird eingestellt, ob das gesamte Unter-Netzwerk direkt an den Linux-Rechner angeschlossen ist, oder ob es noch weiter unterteilt und z.B. durch Bridges (Brücken) verbunden ist. Es macht für gewöhnlich keinen Unterschied, ob man die Standard-Einstellung verwendet oder nicht.

Disable NAGLE algorithm

Der NAGLE-Algorithmus legt fest, wann ein Datagramm losgeschickt werden kann. Die Standard-Einstellung ist auf optimalen Datendurchsatz für die meisten Situationen eingestellt und sollte so belassen werden, da andernfalls die Leistungsfähigkeit der Netzverbindung abnimmt. Da die Verwendung des NAGLE-Algorithmus auch von einem Programm aus über eine Socket-Option verändert werden kann fährt man normalerweise besser, sie standardmäßig zu aktivieren und ihn für Fälle, in denen Programme eine besonders schnelle Reaktionszeit über das Netz benötigen, aus diesem Programm heraus temporär zu deaktivieren.

The IPX protocol

Legt fest, ob das IPX-Protokoll vom Kernel unterstützt werden soll. Es handelt sich dabei um Inter-Netzwerk-Protokoll ähnlich wie IP und ist eines der von Novell verwendeten Protokolle.

Amateur Radio AX.25 Level2

Damit aktiviert man die Unterstützung für das AX.25-Protokoll von Amateur Radio. Dadurch werden eine Reihe zusätzlicher Sockets zur Programmierung freigegeben. Das AX.25-Protokoll wird vorrangig für Packet Radio verwendet.

9 Konfiguration der Netzwerk-Schnittstelle

Ist bis hierher alles glatt gelaufen, besitzt man einen Kernel, der alle Geräte, die man benutzen möchte, unterstützt, außerdem sollten auch alle benötigten Hilfsprogramme vorhanden sein. Der Spaß kann also beginnen. Jedes der gewünschten Geräte muß nun konfiguriert werden. Das bedeutet normalerweise, daß jeder Geräteschnittstelle eine IP-Adresse zugeteilt und das Netzwerk, an das sie angeschlossen ist, mitgeteilt wird.

In den früheren Versionen dieses Textes wurden praktisch sämtliche relevanten Konfigurationsdateien zusammen mit Kommentaren, welche Veränderungen möglich und nötig sind, in vollem Umfang zitiert. Der vorliegende Text geht in dieser Beziehung etwas unterschiedlich vor. Sämtliche Konfigurationsdateien werden von Grund auf neu zusammengestellt, so daß man hinterher genau weiß, was drinnen steht, und vor allem warum. Dies geschieht an den jeweiligen Stellen, an denen diese Dateien benötigt werden.

9.1 Die Device-Einträge in /dev

Die Netzwerk-Implementation unter Linux verwende keinerlei Device-Einträge im Verzeichnis `/dev`. Darin unterscheidet es sich von den Versionen anderer Betriebssysteme. Die benötigten Einträge werden dynamisch im Speicherbereich des Kernels angelegt, und da es sich sowieso nur um Namen handelt gibt es auch keinen Grund, warum sie nach außen hin sichtbar sein sollten. Der Kernel selber bietet alle notwendigen Schnittstellen Kommunikationskanäle, um die Fähigkeiten des Netzwerks effizient zu nutzen.

9.2 Welche Informationen müssen bereitstehen ?

Bevor mit der Konfiguration begonnen werden kann sollte sichergestellt sein, daß einige relevante Daten über die Netzwerk-Verbindung zur Verfügung stehen. Die meisten davon erfährt man vom Netzwerkadministrator oder Internet-Provider.

9.2.1 Die IP Adresse

Dies ist die eindeutige Identifizierungsadresse des Rechners, in der üblichen punktierten Dezimalschreibweise, z.B. `128.253.153.54`. Diese muß man beim Netzwerk-Administrator erfragen.

Wer eine Verbindung über SLIP oder PLIP anstrebt, braucht diese Nummer eventuell nicht. Genaueres steht dann im Abschnitt über SLIP.

Wer nicht plant, eine externe Netzverbindung über SLIP, PPP oder Ethernet aufzubauen und nur das loopback-device verwendet, braucht ebenfalls keine IP-Adresse, da loopback immer die Adresse `127.0.0.1` verwendet.

9.2.2 Die Netz-Maske

Aus Gründen der Leistungsfähigkeit ist es für jedes Netzwerk wünschenswert, die Anzahl der angeschlossenen Rechner möglichst gering zu halten. Deshalb werden große Netzwerke von deren Administratoren oft in kleinere Unterstrukturen aufgeteilt, die man als *subnet* bezeichnet. Die *Network Mask* ist eine Bytemaske, die, wenn man sie mit der IP-Adresse überlagert, die Zugehörigkeit zu einem solchen subnet angibt. Sie ist von essentieller Bedeutung für das Routing in einem Netzwerk, und wer feststellt, daß er zwar problemlos mit Rechnern außerhalb des lokalen Netzwerkes kommunizieren kann, nicht aber mit den lokalen Computern, der hat mit Sicherheit eine falsche Network Mask eingestellt.

Diese Maske wurde normalerweise vom Netzwerk-Administrator bereits bei der Erstellung des Netzwerkes gewählt, er kann deshalb auch diese Information liefern. Die meisten Netzwerke sind Klasse-C Unternetze, die als Netzwerk-Maske den Wert `255.255.255.0` verwenden. Andere, größere Netzwerke (Klasse B) benutzen `255.255.0.0`. Die NET-2/NET-3-Software wählt automatisch eine Standard-Maske aus, wenn einer Schnittstelle eine Adresse zugeordnet wird. Diese Standardeinstellung geht davon aus, daß das Netzwerk **nicht** weiter unterteilt ist. Welche Netz-Maske verwendet wird, hängt von der verwendeten Adresse ab. Im einzelnen gilt folgende Zuordnung:

```
Adressen mit dem ersten Byte:
1-127         255.0.0.0         (Class A)
128-191       255.255.0.0       (Class B)
192+          255.255.255.0     (Class C)
```

Wenn die Standardeinstellung nicht funktioniert, kann man zunächst einfach die anderen Möglichkeiten durchprobieren. Einfacher ist es natürlich, den Administrator oder sonst einen örtlichen Fachman zu fragen.

Lediglich das Loopback-Device benötigt keine Netzwerk-Maske, dasselbe gilt auch für Netzverbindungen via SLIP/PLIP.

9.2.3 Die Netzwerk-Adresse

Die Netzwerkadresse (Network Address) ist die IP-Adresse, durch bitweises `AND` mit der Netmask verknüpft. Ein Beispiel:

```
Netzwerk-Maske:          255.255.255.0
IP-Adresse:              128.253.154.32    &&
                        ----------------
Netzwerk-Adresse:        128.253.154.0     =
```

9.2.4 Die Broadcast-Adresse

Dies ist im Normalfall die Netzwerkadresse, logisch `OR`-verknüpft mit der invertierten Netzwerk-Maske. Man sieht am besten an einem Beispiel, wie das funktioniert:

```
Netzwerk-Maske:              255.255.255.0    !
invertierte Maske:             0.  0.  0.255  =
Netzwerk-Adresse (s.o.):     128.253.154.0    ||
                            -----------------
Broadcast-Adresse:           128.253.154.255  =
```

Aus historischen Gründen verwenden jedoch einige Netzwerke die Netzwerk-Adresse auch als Broadcast-Adresse. Im Zweifelsfall sollte man sicherheitshalber beim Netzwerkadministrator nachfragen.

Wer Zugriff auf einen *sniffer* oder eine ähnliche Möglichkeit, den Datenverkehr auf dem Netzwerk zu überwachen, hat, kann die Broadcast-Adresse eventuell auch durch aufmerksames Verfolgen der Vorgänge auf dem Netz herausbekommen. Hierbei muß man auf Ethernet-Pakete achten, die an die Ethernet-Broadcast-Adresse `ff:ff:ff:ff:ff:ff` adressiert sind. Wenn der Absender eine IP-Adresse aus dem lokalen Netzwerk besitzt und die Protokoll-Identifikation nicht ARP ist, dann handelt es sich vermutlich um eine RIP Routing Anfrage des Routers, die Zieladresse ist dann die lokale Broadcast-Adresse.

Wie gesagt, im Zweifelsfall immer den Netzwerk-Administrator fragen, denn der gibt meist viel lieber einige Ratschläge, als hinterher einen falsch konfigurierten Rechner in seinem Netz zu haben.

9.2.5 Router ('Gateway') Adresse

In fast jedem lokalen Netzwerk gibt es einen Rechner, der dieses Netz mit dem Rest des Internet verbindet. Diesen Rechner bezeichnet man als *Gateway* (Tor) oder Router (Wegbereiter). Für die Vergabe der IP-Adressen der Router gibt es zwei Konventionen: Entweder hat er die höchste Adresse im Netzwerk, oder die kleinste. Da `0` und meist auch `255` keine gültigen Adressen sind (Netzwerk-Adresse und Broadcast-Adresse) sind das `1` oder `254`. Neben dieser Konvention kann der Router aber jede beliebige IP-Adresse haben, ohne daß die Funktionalität des Netzwerkes beeinträchtigt wird, solange die angeschlossenen Rechner diese Adresse kennen. Manche Netzwerke besitzen auch mehrere Router, um das Netz zu entlasten. Deshalb sollte auch hier der Netzwerk-Administrator zu Rate gezogen werden.

Wer das Netzwerk nur im loopback-Modus betreiben will braucht wiederum keine Router-Adresse. Gleiches gilt auch für PPP-Verbindungen, da das PPP-Protokoll diese Adresse automatisch ermittelt. Für SLIP-Verbindungen ist der Router der SLIP-Server und muß beim Provider erfragt werden.

9.2.6 Nameserver-Adressen

Da die IP-Adressen für Menschen eher schlecht zu behalten sind, erhalten Rechner und Netzwerke auch leichter merkbare Namen, z.B. `sunsite.unc.edu`. Um zwischen diese Namen in gültige IP-Adressen zu übersetzen (und umgekehrt) gibt es spezielle *Nameserver*. Die Adresse des nächsten Nameservers erfährt man vom Netzwerk-Administrator. Die andere Möglichkeit besteht darin, auf dem eigenen Rechner einen solchen Nameserver mittels `named` zu konfigurieren. In diesem Fall wäre die Nameserver-Adresse die *Loopback-Adresse* `127.0.0.1`. Näheres dazu steht im Abschnitt 'named'. Es können auch mehrere Nameserver-Adressen konfiguriert werden, falls einer nicht erreichbar ist.

Für Loopback benötigt man keinen Nameserver, da man sich sowieso nur mit dem eigenen Rechner unterhält.

9.2.7 Hinweis für SLIP/PLIP/PPP Nutzer

Viele der obengenannten Informationen können für SLIP/PPP/PLIP-Benutzer gar nicht notwendig sein. Das hängt davon ab, wie die Netzwerk-Verbindung genau hergestellt wird, und über welche Fähigkeiten der Rechner am anderen

Ende der Verbindung besitzt. Darauf wird in den entsprechenden Abschnitten zur Konfiguration von SLIP/PPP und PPP eingegangen.

9.3 /etc/rc.d/rc.inet1,2

Je nach Distribution können diese beiden Dateien auch in einer einzigen mit dem Namen `/etc/rc.net` oder `/etc/init.d/network` zusammengefaßt sein

Prinzipiell könnten alle Kommandos zur Konfiguration des Netzwerkes auch von Hand in der Kommandozeile eingegeben werden, um das Netzwerk zu starten. Gerade bei dauerhaften Verbindungen wie Ethernet ist es aber wünschenswert, daß der Rechner automatisch beim Einschalten diese Konfiguration durchführt.

Die `rc`-Dateien (kurz für *runtime command*) sind genau für solche Zwecke vorgesehen. Für diejenigen, die sich nicht so genau mit Unix auskennen: Bei diesen Dateien handelt es sich um Shell-Scripts, die beim Systemstart vom *init*-Programm ausgeführt werden. die `rc`-Dateien starten u.a. die grundlegenden Systemprogramme wie `syslog`, `update` oder `cron`. Sie sind das Analogon zur Datei `autoexec.bat` bei MS-DOS. Diese Dateien residieren unterhalb des `/etc`-Verzeichnisses. Der Linux Filesystem Standard legt aber nicht exakt fest, wo diese Dateien stehen müssen. Es können entweder die BSD-Konventionen (`/etc/rc.*`) oder die System-V-Konventionen (`/etc/rc.d/rc.*`) realisiert sein. Da die System-V Version etwas übersichtlicher ist (alle Dateien stehen in einem Verzeichnis) und sie außerdem auch von den Entwicklern (Alan, Fred) verwendet wird, wird im folgenden diese Konvention verwendet. Das bedeutet, daß die Netzwerk-Dateien sich in `/etc/rc.d` befinden und die Namen `rc.inet1` und `rc.inet2` besitzen. Teilweise gibt es eine Datei `/etc/rc`, die dann die einzelnen Programme in `/etc/rc.d` startet, manchmal ist auch init bereits so kompiliert, daß es selber in `/etc/rc.d` sucht. Wo die Dateien stehen ist im Prinzip egal, solange `init` sie findet.

```
System-V               Debian und andere
==================     ========================
/etc/rc.d/rc.inet1     /etc/init.d/network

/etc/rc.d/rc.inet2     /etc/init.d/netbase
                       /etc/init.d/netstd_init
                       /etc/init.d/netstd_nfs
                       /etc/init.d/netstd_misc
```

Es ist nicht wichtig, wo genau die Konfigurationsbefehle für die Schnittstellen auftauchen, solange dies geschieht, bevor Netzwerk-Dämonen oder -Anwendungen gestartet werden.

Im weiteren werden die Netzwerk-Dateien als `rc.inet1` und `rc.inet2` im Verzeichnis `/etc/rc.d` referiert. Wer also eine Distribution verwendet, die diese Daten an einer anderen Stelle verwaltet, der muß jeweils selbst nachschauen, wo die besprochenen Optionen eingestellt werden und sie dort entsprechend anpassen.

9.3.1 rc.inet1

Die Datei `rc.inet1` konfiguriert das TCP/IP-Basissystem. Dazu werden die beiden Programme `/sbin/ifconfig` und `/sbin/route` benutzt .

ifconfig

`/sbin/ifconfig` konfiguriert die Schnittstelle mit allen nötigen Parametern, die zum Betrieb notwendig sind, also IP-Adresse, Netzwerk-Maske, Broadcast-Adresse usw. Der Befehl `ifconfig` ohne weitere Parameter zeigt die Konfiguration aller existierenden Netzwerk-Devices an. Die Online-Hilfe zu ifconfig (8) gibt weitere Informationen zur Benutzung.

route

Das Programm /sbin/route erzeugt und verwaltet eine Tabelle (die Routing-Table), in der eingetragen ist, welche IP-Adressen über welche Schnittstellen erreicht werden können. Der Netzwerk-Code im Kernel schaut für jedes zu sendende Datagramm in dieser Tabelle nach, über welchen Weg (Route) das Paket zugestellt werden soll. Ohne Parameter gestartet, zeigt route den Inhalt der momentanen Routing-Table an. Näheres steht in der Online-Hilfe.

9.3.2 rc.inet2

In rc.inet2 werden alle benötigten Netzwerk-Dämonen gestartet, also Programme wie inetd oder portmapper. Auf die Einzelheiten dieser Programme wird weiter unten genauer eingegangen, die Datei ist hier nur erwähnt, um die Aufgabentrennung zwischen rc.inet1 und rc.inet2 darzulegen. Die Trennung in zwei Dateien 1 und 2 zeigt auch deutlicher die Notwendigkeit, die Schnittstellen zu konfigurieren, **bevor** Netz-Dämonen oder -Anwendungen gestartet werden. Dies ist besonders wichtig für diejenigen, die nur eine einzige rc.net-Datei besitzen, um zu verstehen, was in der zweite Hälfte dieser Datei geschieht.

9.4 Die Konfiguration des Loopback-Device (immer)

Das Loopback-Device ist keine wirkliche Hardware. Es ist eine reine Software-Implementation, die sich aber wie eine physikalische Schnittstelle verhält. Sie ermöglicht es dem Rechner, mit sich selbst zu kommunizieren und erlaubt es, Netzwerk-Programme zu testen, ohne wirklich an ein Netzwerk angeschlossen zu sein. Die ist von großem Vorteil wenn man eine nicht dauerhafte SLIP-Verbindung hat oder selber Netzwerk-Software schreibt.

Das Loopback-Device verwendet per Konvention **immer** die IP-Adresse 127.0.0.1, obwohl beliebige Adressen eingestellt werden können. Diese wird deshalb bei der Konfiguration angegeben.

Unter Linux trägt das Loopback-Device den Namen lo. Der erste Eintrag in der Datei rc.inet1 sieht also so aus:

```
#!/bin/sh
#
# rc.inet1   --  configures network devices.
#
# Attach the loopback device.
/sbin/ifconfig lo 127.0.0.1
#
# Add a route to point to the loopback device.
/sbin/route add 127.0.0.1
# End loopback
#
```

Hierbei wurde ifconfig werwendet, um der Loopback-Schnittstelle die IP-Adresse zuzuweisen, und route, um in der Routing-Table einen Eintrag zu erzeugen der sicherstellt, daß alle Datagramme an die Adresse 127.0.0.1 über die Loopback-Schnittstelle gesandt werden.

Zwei sehr wichtige Dinge müssen hier noch erwähnt werden:

Erstens, die Netzwerk-Maske und Broadcast-Adresse wurden beim ifconfig-Befehl nicht angegeben, deshalb werden automatisch die Standard-Werte verwendet. Diese werden mit allen anderen Schnittstellenparametern angezeigt, wenn man ifconfig ohne Parameter aufruft:

```
% ifconfig
lo        Link encap Local Loopback
          inet addr 127.0.0.1  Bcast 127.255.255.255  Mask 255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU 2000  Metric 1
```

```
              RX packets 0 errors 0 dropped 0 overrun 0
              TX packets 30 errors 0 dropped 0 overrun 0
    %
```

Zweitens, es ist aus dem angegebenen Befehl nicht ersichtlich, woher `route` weiß, daß als Schnittstelle für die Adresse `127.0.0.1` das Loopback-Device verwendet werden soll. Aber da es sich um einen Standard-Wert handelt, schließt das `route`-Programm aus Adresse und Netzwerk-Maske automatisch, daß es sich um die Loopback-Adresse handelt. Die so erstellte Routing-Table kann durch den Befehl `route` ohne Parameter angezeigt werden:

```
% route
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       *               255.0.0.0       U     0      0       30 lo
%
```

Achtung: Es empfiehlt sich, `route` hierfür mit der Option `-n` zu verwenden, wenn der Name-Resolver (gegenseitige Zuordnung von Namen und IP-Adressen) noch nicht korrekt konfiguriert ist. Diese Option bewirkt dann, daß lediglich die IP-Adresse angezeigt wird und nicht versucht wird, diese Adressen in Rechnernamen zu übersetzen.

9.5 Die Konfiguration des Ethernet-Device (optional)

Dieser Abschnitt ist nur von Interesse für denjenigen, der eine Ethernet-Karte in seinem Rechner betreiben will. Alle anderen können ihn überspringen.

Die Konfiguration einer Ethernet-Karte ist nur geringfügig komplizierter als die des Loopback-Device, da Netzwerk-Maske und Broadcast-Adresse mitangegeben werden sollten. Die Standard-Werte sollten nur verwendet werden wenn man auch sicher ist, daß diese für das lokale Netzwerk auch korrekt sind.

Benötigt werden also die zugeteilte IP-Adresse, die lokale Netzwerk-Maske und die Broadcast-Adresse.

Das erste Ethernet-Device in einem Linux-System trägt die Bezeichnung `eth0`, das zweite `eth1` und so weiter. Der folgende Code-Abschnitt muß in der Datei `rc.inet1` angefügt werden, damit die Karte richtig eingetragen wird. Die IP-Adresse muß dabei natürlich durch die eigene ersetzt werden:

```
#
# Attach an ethernet device
#
#  configure the IP address, netmask and broadcast address.
/sbin/ifconfig eth0 IPA.IPA.IPA.IPA
/sbin/ifconfig eth0 netmask NMK.NMK.NMK.NMK
/sbin/ifconfig eth0 broadcast BCA.BCA.BCA.BCA
#
# add a network route to point to it:
/sbin/route add -net NWA.NWA.NWA.NWA device eth0
#
# End ethernet
#
```

Dabei bezeichnet

IPA.IPA.IPA.IPA
: die IP-Adresse,

NMK.NMK.NMK.NMK
: die Netzwerk-Maske,

BCA.BCA.BCA.BCA

die Broadcast-Adresse und

NWA.NWA.NWA.NWA

die Netzwerk-Adresse.

Die Option `-net` teilt dem `route`-Kommando mit, daß es sich bei der angegebenen Adresse um ein hinzuzufügendes *Netzwerk* und nicht um einen einzelnen *Host* (Rechner) handelt. Die Option kann entfallen, wenn die zu konfigurierenden Netzwerke in der Datei `/etc/networks` eingetragen sind, `route` nutzt dann diese Datei, um Netzwerkadressen automatisch zu erkennen. Das Format dieser Datei wird später in einem eigenen Abschnitt erläutert.

9.6 Die Konfiguration eines SLIP-Device (optional)

SLIP (Serial Line Internet Protocoll) erlaubt es, TCP/IP über eine serielle Datenleitung zu benutzen. Dies kann eine Telefonverbindung über ein Modem sein oder eine dauerhafte, z.B. gemietete Standleitung. Natürlich benötigt man für SLIP einen zweiten Rechner als Gegenstelle, den *SLIP-Server*. Derartige SLIP-Server werden von vielen Universitäten und inzwischen auch von kommerziellen Anbietern bereitgestellt.

SLIP nutzt die seriellen Ports eines Rechners, um IP-Datagramme weiterzuleiten. Hierzu muß es die Kontrolle über das serielle Device übernehmen. Die SLIP-Devices tragen die Bezeichnungen `sl0`, `sl1` usw. Wie sind diese nun den seriellen Devices zugeordnet? Der Netzwerk-Code benutzt dazu einen *ioctl* (i/o control), um die seriellen Devices in SLIP-Devices umzuändern. Es werden zwei Programme zur Verfügung gestellt, die dies bewerkstelligen können: `dip` und `slattach`.

9.6.1 dip

`dip` (Dialup IP) ist ein komfortables und sehr leistungsfähiges Programm, das sämtliche notwendigen Schritte ausführen kann, um einen Rechner über SLIP an das Internet anzuschließen: Setzten der Kommunikationsgeschwindigkeit auf dem seriellen Port, Herstellen der Modem-Verbindung, automatischer Login, Extrahieren von notwendigen Informationen wie IP-Nummern für dynamische Adressvergabe aus den Rückmeldungen des Servers und Ausführung des *ioctl*, um die serielle Leitung auf den SLIP-Modus zu schalten. Dies alles kann durch eine leistungsfähige Script-Sprache einfach konfiguriert und vollautomatisch durchgeführt werden.

Die Weiterentwicklung von `dip` verläuft mittlerweile getrennt von restlichen Netzwerk-Code und ist deshalb nicht mehr bei den *net-tools* enthalten. Man muß es sich also separat besorgen. Es gibt inzwischen verschiedene Versionen von `dip`, die eine Vielzahl an zusätzlichen neuen Optionen und Möglichkeiten bieten. Es lohnt sich durchaus, die einzelnen Versionen anzusehen und dann diejenige zu verwenden, die den eigenen Wünschen am besten entspricht. Am weitesten verbreitet scheint aber die Version `dip-uri` zu sein, deshalb werden die Beispiele im Text auf dessen derzeitigen Version basieren.

`dip-uri` findet man auf:

```
sunsite.unc.edu:/pub/Linux/system/Network/serial/dip337o-uri.tgz
```

Um es zu installieren, geht man folgendermassen vor:

```
% cd /usr/src
% gzip -dc dip337o-uri.tgz | tar xvf -
% cd dip.3.3.7o

<Makefile editieren>

% make install
```

Das Makefile geht davon aus, daß eine Group namens `uucp` existiert, unter deren Group-ID die Programme installiert werden. Wer will, kann dies je nach lokaler Konfiguration in `dip` oder `SLIP` umändern.

9.6.2 slattach

Im Gegensatz zu `dip` ist `slattach` ein recht spartanisches Programm, das zwar sehr einfach zu bedienen ist, aber bei weitem nicht die Möglichkeiten von `dip` bietet. Es hat keinerlei Script-Fähigkeiten und wird vollständig über die Kommandozeile gesteuert. Seine einzige Aufgabe ist es, den seriellen Port für SLIP zu konfigurieren. Dabei geht es davon aus, daß bereits eine serielle Verbindung zum anderen Rechner besteht und alle notwendigen Parameter bekannt sind, da es keine Standardwerte gibt. Dadurch eignet sich `slattach` ganz besonders für dauerhafte Verbindungen zum Server, sei es über ein (Null-Modem) Kabel oder eine Standleitung.

9.6.3 Und wann benutze ich welches Programm ?

`dip` ist das Programm der Wahl wenn die SLIP-Verbindung über ein Modem hergestellt wird. `slattach` hingegen ist für dauerhafte Verbindungen besser geeignet, wenn zu deren Herstellung keine spezielle Aktionen notwendig sind. Im Abschnitt 'Permanente SLIP Verbindung' werden hierzu genauere Details gegeben.

Die Konfiguration des SLIP-Device verläuft fast genauso so wie für das Ethernet-Device (siehe den vorigen Abschnitt). Ein paar entscheidende Unterschiede gibt es dennoch.

Zunächst gibt es im Gegensatz zu einem Ethernet-Netzwerk bei SLIP nur zwei Rechner, einen an jedem Ende der Verbindung. Anders als beim Ethernet besteht auch nicht sofort eine Netzwerkverbindung, sobald die Rechner physikalisch verbunden sind, vielmehr sind, je nach Art der Verbindung, zusätzliche Initialisierungen notwendig.

Bei der Verwendung von `dip` geschieht dies normalerweise nicht beim Systemstart sondern zu einem späteren Zeitpunkt, wenn die Verbindung auch wirklich benutzt werden soll. Der gesamte Vorgang läßt sich aber auch in diesem Fall vollständig automatisieren. (Dauerhafte) Verbindungen, die über `slattach` konfiguriert werden, werden meist bereits beim Systemstart aufgebaut und die notwendigen Befehle in die Datei `rc.inet1` aufgenommen. Dies wird später beschrieben.

Es gibt zwei unterschiedliche Arten von SLIP-Servern: Solche, die statische IP-Adressen benutzen und solche, die die IP-Adressen dynamisch vergeben. Für gewöhnlich meldet sich ein SLIP-Server zunächst mit einer Login-Meldung und fragt dann nach Benutzername und Paßwort. `dip` kann in beiden Fällen den Login-Prozeß automatisch durchführen.

9.6.4 Dip bei statischem SLIP-Server über Modem-Verbindung

Ein statischer SLIP-Server vergibt an jeden Benutzer eine eigene IP-Adresse. Bei jedem Verbindungsaufbau wird der SLIP-Server, basierend auf dem Login-Namen, eine bestimmte IP-Adresse verwenden und alle Datagramme, die an diese Adresse sind, über die serielle Leitung an den eigenen Rechner weiterleiten. Da sich somit die IP-Adresse niemals ändert, kann sie dauerhaft in der Datei `/etc/hosts` eingetragen werden. Auch in den Dateien `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` und `rc.local` kann die Adresse dann an entsprechenden Stellen fest eingetragen werden. In `rc.inet1` sind wie bereits gesagt keine Einträge betreffend das SLIP-Device nötig, da alle Konfiguration von `dip` übernommen wird. Die Netzwerk-Parameter müssen deshalb `dip` mitgeteilt werden.

Benutzer statischer SLIP-Server können direkt im Abschnitt 'Die Benutzung von dip' weiterlesen.

9.6.5 Dip bei dynamischem SLIP-Server über Modem-Verbindung

Ein *dynamischer* SLIP-Server verteilt die IP-Adressen nicht anhand der Benutzer-ID, sondern weist jedem, der sich einlogged eine gerade freie Nummer aus einem Pool von Adressen zu. Es gibt also keine Garantie, daß man jedesmal eine bestimmte IP-Nummer zugewiesen bekommt, außerdem kann (und wird) dieselbe Nummer einem anderen zugeteilt werden, nachdem die Verbindung beendet ist. Bei jedem Verbindungsaufbau wird so die nächste freie IP-Adresse aus dem vom Administrator des SLIP-Servers eingestellten Bereich herausgesucht und dann dieser einen Verbindung zugeordnet. Nach der normalen Login-Prozedur wird der SLIP-Server diese Adresse zusammen mit der Login-Meldung mitteilen und ab diesem Moment alle Datagramme an diese Adresse über die Modem-Verbindung weiterleiten.

Die Konfiguration für diesen Typ von Server unterscheidet sich kaum von demjenigen für statische Server. Es muß lediglich ein Schritt eingefügt werden, bei dem die zugeteilte Adresse aus der Rückmeldung des Servers herausgesucht wird, um danach wie beim statischen Server das SLIP-Device damit zu konfigurieren.

Auch hier verrichtet `dip` die Hauptarbeit, neuere Versionen sind sogar schlau genug, selbständig die IP-Adresse in der Willkommens-Meldung des Servers zu finden und für die Konfiguration des SLIP-Device zu verwenden.

Auch für Benutzer von dynamischen SLIP-Servern ist das wichtigste also die richtige Konfigurierung von `dip`, was im folgenden Abschnitt beschrieben wird.

9.6.6 Die Benutzung von dip

Wie bereits dargestellt handelt es sich bei `dip` um ein äußerst leistungsfähiges Programm, das den Verbindungsaufbau stark vereinfachen, ja sogar automatisieren kann. `dip` übernimmt dabei das Anwählen des SLIP-Servers, die gesamte Login-Kommunikation, es konfiguriert den seriellen Port für das SLIP-Protokoll und führt die nötigen `ifconfig` und `route` Befehle aus, um das SLIP-Device richtig zu konfigurieren.

Die 'Konfiguration' von `dip` besteht dabei darin, ein *dip-Script* zu erstellen. Dieses besteht aus einer Reihe von Befehlen, die `dip` versteht, und die angeben, welche Aktionen ausgeführt werden sollen. Das `dip`-Paket kommt mit einem Beispiel für eine solche Script-Datei, dieses hat den Namen `sample.dip`. An ihm kann man sehr gut den Aufbau einer solchen Script-Datei erkennen und lernen, wie alles abläuft. `dip` ist ein sehr leistungsfähiges Programm mit zu vielen Optionen, als daß diese hier alle erläutert werden könnten. Man sollte die Online-Hilfe lesen, um sich über den ganzen Umfang an Möglichkeiten zu informieren und die README-Dateien und Beispiele ansehen, die zusammen mit `dip` geliefert werden.

Wer sich das Beispiel `sample.dip` ansieht wird feststellen, daß es von einem statischen SLIP-Server ausgeht, also die Kenntnis der IP-Nummer vorausgesetzt wird. Die neueren Versionen von `dip` umfassen aber auch ein Kommando, um bei dynamischen SLIP-Servern die zugeteilte IP-Adresse zu erkennen und entsprechend zu konfigurieren. Das folgende Beispiel ist eine modifizierte Version des `sample.dip` aus *dip337o-uri.tgz*. Es stellt vermutlich eine gute Ausgangsbasis für die meisten Anwendungen dar. Es empfiehlt sich, dieses Script z.B. nach `/etc/dipscript` zu kopieren und dann den eigenen Anforderungen entsprechend anzupassen.

```
#
# sample.dip    Dialup IP connection support program.
#
#       Diese Datei zeigt (soll zeigen) wie man DIP benutzt.
#       Es sollte mit den gaengigen dynamischen Servern funktionieren.
#       Wer seinen Zugang ueber einen statischen Server hat, sollte
#       die Standard-Version von sample.dip aus dem dip-Paket
#       verwenden.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Festlegen von Name und Adresse der Gegenseite
```

```
# Mein Verbindungs-Rechner ist 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Setze die Netzmaske f\"{u}r sl0 auf 255.255.255.0
netmask 255.255.255.0
# Setze den gewuenschten seriellen Port und dessen Geschwindigkeit
port cua02
speed 38400

# Reset fuer das Modem und das Terminal.
# Dies verursacht evtl Probleme bei manchen Leuten!
reset

# Die vordefinierten "Errlevel" sind:
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# Sie koennen geaendert werden, dazu sucht man (mit grep) in den
# Quelldateien (*.c) nach "addchat()"

# Vorbereitung zum Waehlen
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# OK, die Verbindung steht.  Jetzt der Login
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# Jetzt sind wir eingelogged
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Setze den Server in den SLIP-Modus
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Finde die vom Server zugeteilte IP-Adresse.
#  Dabei wird davon ausgegangen, da{\ss} der Server, nachdem er in den
#  SLIP-Modus uebergewechselt ist, diese Adresse ausgibt.
get $locip remote 30
if $errlvl != 0 goto prompt_error
```

```
# Lege die SLIP-Parameter fest
get $mtu 296
# Dies stellt sicher, da{\ss} der Befehl
# "route add -net default xs4all.hacktic.nl" ausgefuehrt wird.
default

# Eine Meldung auf die Console, da{\ss} alles funktioniert, und dann wird
# die eigene Seite in den (C)SLIP-Modus geschaltet.
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT beim Warten auf den SLIP-Start
goto error

login_trouble:
print Probleme beim Warten auf den Login: prompt...
goto error

password:error:
print  Probleme beim Warten auf den Password: prompt...
goto error

modem_trouble:
print  Probleme mit dem Modem...
error:
print CONNECT mit $remote GESCHEITERT
quit

exit:
exit
```

Dieses Beispiel geht von einem *dynamischen* SLIP-Server aus. Beim Zugang über einen *statischen* SLIP-Server genügt die originale Datei `sample.dip` aus dem `dip337o-uri.tgz` Paket.

Die interessante Stelle in diesem Script ist der Befehl `get $local`. Daraufhin untersucht `dip` allen eingehenden Text auf eine Zeichenkette, die wie eine IP-Adresse aussieht, also Ziffern, die durch Punkte '.' getrennt sind. Diese Spezialität wurde speziell wegen der Zunahme von dynamischen Servern eingeführt, damit auch für diese der Verbindungsaufbau automatisch und ohne Eingreifen des Benutzers durchgeführt werden kann.

Das obige Beispiel legt automatisch die Default-Route auf die SLIP-Verbindung. Wer das nicht braucht, weil er z.B. eine Ethernet-Verbindung als Default besitzt, muß im Script den `default` Befehl entfernen. Nachdem das Script erfolgreich abgearbeitet wurde zeigt ein `ifconfig`, daß jetzt ein Device `sl0` existiert. Dies ist das SLIP-Device. Seine Konfiguration kann, wenn dies notwendig ist, manuell verändert werden. Hierfür müssen die geeigneten `ifconfig` und `route` Befehle benutzt werden.

Es muß noch darauf hingewiesen werden, daß `dip` eine ganze Reihe von unterschiedlichen Protokollen unterstützt, die mit dem `mode` Befehl aktiviert werden können. Die gängigste Variante ist *cSLIP*, das ist SLIP mit Komprimierung. Es ist zwingend notwendig, daß beide Seiten dieselbe Einstellung verwenden, da sonst keine Kommunikation möglich ist. Das vom Server verwendete Protokoll muß deshalb beim Provider erfragt werden.

Das aufgeführte Beispiel ist ziemlich stabil und fängt einige mögliche Fehler ab. Die Online-Hilfe `man dip` gibt weitere Hinweise dazu. Natürlich kann das Script auch dahingehend erweitert werden, daß z.B. im Falle einer mißlungenen Verbindungsaufnahme ein erneuter Wählversuch gestartet wird, oder daß nacheinander verschiedene Server probiert werden, bis es zu einer Erfolgreichen Verbindung kommt.

9.6.7 Permanente SLIP Verbindung (Standleitung) mit slattach

Wer ein Verbindungskabel zwischen zwei Rechnern hat oder sogar das Glück, eine Standleitung oder eine ähnliche dauernde serielle Verbindung zu besitzen, der braucht sich nicht mit `dip` zu befassen, um die Verbindung herzustellen. `slattach` ist ein sehr einfach zu bedienendes Programm, welches gerade genug Funktionalität bietet, um eine solche Verbindung aufzubauen.

Da diese Verbindung eine permanente sein soll ist es angebracht, die nötigen Befehle in die Datei `rc.inet1` aufzunehmen. Kurz gesagt muß man nur sicherstellen, daß das serielle Device die korrekte Geschwindigkeit eingestellt hat und es danach in den SLIP-Modus umschalten. `slattach` führt diese beiden Dinge mit einem einzigen Befehl aus. Dafür müssen die folgenden Zeilen zum bestehenden `rc.inet1` **hinzugefügt** werden:

```
#
# Attach a leased line static SLIP connection
#
#  configure /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End static SLIP.
```

Wie gehabt müssen für

IPA.IPA.IPA.IPA
: die eigene IP Adresse und für

IPR.IPR.IPR.IPR
: die IP Adresse des anderen Rechners

angegeben werden.

`slattach` teilt dem angegebenen seriellen Device das erste freie SLIP-Device zu, im angegeben Fall also `sl0`. Dieses wird dann im folgenden *ifconfig* konfiguriert. Ein weiteres `slattach`-Kommando für eine zusätzliche Leitung würde dieser dann das Device `sl1` zuteilen.

Auch `slattach` kennt unterschiedliche Protokolle, die mit dem `-p` Schalter aktiviert werden können. Meist wird hier `SLIP` oder `cSLIP` verwendet, je nachdem, ob Kompression eingeschaltet werden soll. Wie auch bei `dip` gilt: Beide Seiten der Verbindung müssen dasselbe Protokoll konfigurieren, sonst geht nichts!

9.7 Die Konfiguration des PLIP Device (optional)

PLIP (Parallel Line IP) dient, wie auch SLIP, zum Aufbau einer *point to point* Netzwerk-Verbindung, die also genau zwei Punkte miteinander verbindet. Im Unterschied zu SLIP, welches die seriellen Ports verwendet, ist PLIP aber dazu gedacht, diese Verbindung über die parallele Schnittstelle des Rechners zu verwirklichen. Da über eine parallele Schnittstelle mehr als ein Bit gleichzeitig übertragen werden kann, können über eine PLIP-Verbindung höhere Übertragungsraten erreicht werden als über eine normale serielle Leitung. Hierfür kann auch die billigste parallele Schnittstelle, der Drucker-Port, verwendet werden. Ähnliche Geschwindigkeiten sind für SLIP nur bei Verwendung der recht teuren 16550AFN UART's in den seriellen Schnittstellen möglich.

Unerfreulicherweise verwenden einige Notebooks Chipsätze, die nicht mit PLIP zusammenarbeiten, da einige Kombinationen von Signalen, die PLIP unbedingt benötigt, nicht erlaubt sind, da sie von Druckern nicht verwendet werden.

Das PLIP-Interface von Linux ist kompatibel zum *Crynwyr Packet Driver PLIP*. Das bedeutet, daß ein Linux-Rechner via PLIP auch mit einem DOS-Rechner verbunden werden kann.

Der PLIP-Treiber kann auf zwei unterschiedliche Weisen eingebunden werden, er kann entweder dauerhaft in den Kernel einkompiliert oder unter Verwendung des *modules* Paketes bei Bedarf hinzugeladen werden. Da PLIP meist eine dauerhafte Verbindung darstellt empfiehlt sich meist die Variante, den Treiber fest in den Kernel einzubinden.

Bei der Kernel-Kompilation gibt es nur eine einzige Datei, die man eventuell ansehen muß. Diese ist `/usr/src/linux/driver/net/CONFIG`, und sie enthält die `plip` Timer in mS. Die Standardeinstellungen sind aber in den meisten Fällen ausreichend. Nur auf besonders langsamen Computern muß der eingestellte Wert unter Umständen vergrößert werden, wobei dadurch tatsächlich der Timer des anderen Computers beeinflußt wird.

Der Treiber nimmt folgende Standard-Werte an:

```
device  i/o addr    IRQ
------  --------    -----
plip0   0x3BC           5
plip1   0x378           7
plip2   0x278           2 (9)
```

Wessen parallele Schnittstelle **nicht** mit einer dieser Einstellungen übereinstimmt, der kann den IRQ eines Ports mit der `irq`-Option des `ifconfig`-Befehles verändern. Eventuell müssen auch die IRQs für die Druckerschnittstelle im ROM BIOS freigegeben werden, falls dieses das unterstützt.

Um das `plip`-Device zu konfigurieren, müssen die folgenden Zeilen zur Datei `rc.inet1` hinzugefügt werden:

```
#
# Attach a PLIP interface
#
#  configure first parallel port as a plip device
/sbin/ifconfig plip0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End plip
```

Auch hier gilt:

IPA.IPA.IPA.IPA
: ist die eigene IP Adresse.

IPR.IPR.IPR.IPR
: ist die IP Adresse des anderen Rechners.

Der Parameter *pointopoint* hat dieselbe Bedeutung wie bei der Konfiguration des SLIP-Device, es gibt die Adresse des Rechners am anderen Ende der Leitung an.

In fast allen Belangen kann man das `plip` Interface genauso behandeln als wenn es sich um ein *SLIP* Interface handeln würde. Lediglich die Verwendung von `dip` ist weder nötig noch möglich.

9.7.1 Diagramm eines PLIP-Kabels

PLIP wurde so ausgelegt daß die Kabel dieselbe Pin-Belegung aufweisen wie diejenigen, die von den meisten DOS-Programmen zur Datenübertragung zwischen zwei PCs verwendet werden.

Dieses Pin-Belegungs-Schema sieht folgendermaßen aus (es ist der Datei `/usr/src/linux/drivers/net/plip.c` entnommen):

```
Pin Name     Connect pin - pin
---------    ------------------------------
GROUND       25 - 25
D0->ERROR    2 - 15
ERROR->D0    15 - 2
D1->SLCT     3 - 13
SLCT->D1     13 - 3
D2->PAPOUT   4 - 12
PAPOUT->D2   12 - 4
D3->ACK      5 - 10
ACK->D3      10 - 5
D4->BUSY     6 - 11
BUSY->D4     11 - 6
D5           7*
D6           8*
D7           9*
STROBE       1*
FEED         14*
INIT         16*
SLCTIN       17*
```

Achtung: Pins, die mit einem Stern '*' markiert sind, dürfen nicht verbunden werden. 18,19,20,21,22,23 und 24 sind zusätzliche GROUND-Anschlüsse.

Werden geschirmte Kabel verwendet, sollte die Abschirmung **nur auf einer Seite** mit dem Metall des Steckers verbunden werden

Warnung! Falsch verdrahtete PLIP-Kabel können die Controller-Karte zerstören. Also lieber einmal zuviel überprüfen, daß alle Pins korrekt verbunden sind, das erspart unnötigen Ärger.

Obwohl PLIP-Kabel im Prinzip sehr lang sein können, sollte man es nach Möglichkeit vermeiden. Die Spezifikationen erlauben Kabellängen von etwa einem Meter. Aber je länger ein Kabel ist, desto empfindlicher reagiert es auf Störungen durch elektromagnetische Felder wie sie z.B. durch Blitze, Stromkabel oder Radiosender verursacht werden. Im Extremfall kann durch solche Fremdeinstrahlungen sogar der Controller in Mitleidenschaft gezogen werden. Wer seine Rechner unbedingt über eine längere Strecke miteinander vernetzen will sollte sich wirklich überlegen, ob er sich nicht doch lieber zwei Netzwerk-Karten und ein koaxiales Kabel zulegen sollte.

10 Routing

Nachdem jetzt alle Netzwerk-Devices konfiguriert sind, muß man sich Gedanken darüber machen, auf welchen Wegen der Computer die Datagramme weiterleitet. Ist überhaupt nur ein einziges Device vorhanden ist diese Frage sehr leicht zu beantworten: Alle Datagramme, die nicht an den eigenen Rechner adressiert sind, müssen über diese Schnittstelle transferiert werden. Bei mehr als einer Schnittstelle wird es aber komplizierter. Hier muß man bei jedem Datagramm nachsehen, über welche Schnittstelle der Rechner, an den das Datagramm adressiert ist, angeschlossen ist. Dieses 'Routing' ist im Prinzip aber recht einfach, auch wenn es manchem auf den ersten Blick nicht so erscheinen mag.

Den Inhalt der momentanen Routing-Tabelle kann man sich ansehen, indem man das `route`-Kommando ohne Parameter aufruft.

Unter Unix gibt es vier unterschiedliche Routing- Mechanismen, auf die im folgenden kurz eingegangen werden soll.

10.1 Statisches Routing

Wie der Name bereits andeutet handelt es sich dabei um ein quasi 'fest verdrahtetes' Routing. Das bedeutet, daß der angegebene Weg unter allen Umständen benutzt wird, auch wenn zu einem späteren Zeitpunkt eine bessere Verbindung zur Verfügung stehen würde, oder wenn die angegebene Verbindung zusammenbricht. Auch wenn noch eine alternative Verbindung bestehen würde, sie würde bei statischem Routing nicht benutzt. Aus diesem Grunde werden sie normalerweise nur für sehr einfache Netze verwandt, wenn es sowieso keine weitere Verbindungsmöglichkeit zum anderen Rechner gibt.

Unter Linux wird dieses statische Routing insbesondere für SLIP- und PLIP-Verbindungen angelegt, wenn der Befehl `ifconfig` mit dem Parameter `pointopoint` verwendet wird. Der entsprechende `route`-Befehl, um eine solche statische Verbindung in die Routing-Tabelle einzutragen, lautet:

```
%/sbin/route add IPR.IPR.IPR.IPR
```

IPR.IPR.IPR.IPR

ist dabei die Adresse des Rechners am anderen Ende der Leitung.

10.2 Default Route

Für die meisten Rechner, die als Benutzer-Workstations im Netzwerk angeschlossen sind und keine speziellen Aufgaben in diesem Netzwerk wahrnehmen ist die Deklaration einer *Default Route* das Standardvorgehen. Eine solche Route ist eine besondere Art von statischer Route, die für alle Adressen angewandt wird, für die nicht explizit eine andere Route angegeben ist.

Wer also nur über eine einzige Netzwerk-Schnittstelle verfügt, sei es nun eine SLIP-Verbindung oder eine Ethernet-Karte, der sollte seine Default Route auf dieses Device einrichten. Dazu ist weiterhin eine Adresse notwendig, diejenige des *Routers*. Dies läßt sich am besten am Beispiel eines (lokalen) Ethernet-Netzwerkes sehen: Soll ein Datagramm an einen Rechner innerhalb des lokalen Netzes gesandt werden, so kann dies der Kernel selständig erkennen (er macht dies anhand der bereits erläuterten Netzwerk-Maske). Soll das Datagramm aber an einen Rechner außerhalb dieses Netzes gehen, so benötigt der Kernel die Adresse eines anderen Rechners, an den er das datagramm weiterreichen kann, und der dann die weitere Zustellung übernimmt. Einen solchen Rechner bezeichnet man als *Router* oder *Gateway*. Im Falle eines Internet-Zuganges über SLIP übernimmt der *SLIP Server* diese Aufgabe, seine IP-Adresse muß dann als Router eingetragen werden. In lokalen Ethernet-Netzen erfährt man die Adresse des Routers vom Systemadministrator.

Um nun die Default Route festzulegen, müssen die folgenden Zeilen in der Datei `rc.inet1` **nach** der Konfiguration sämtlicher Netzwerk-Devices eingefügt werden:

```
#
# Add a default route.
#
/sbin/route add default gw RGA.RGA.RGA.RGA
#
```

RGA.RGA.RGA.RGA

ist die Router/Gateway Adresse.

10.3 Proxy ARP

Diese Methode ist unschön, birgt einige Gefahren und sollte nur unter besonderer Vorsicht verwendet werden. Da aber manche darauf schwören, soll es hier nicht unerwähnt bleiben.

Bei weitem das größte Bedürfnis für *Proxy ARP* besteht bei all denjenigen, die einen Linux-Rechner als SLIP-Server konfigurieren wollen. Soll hingegen ein PPP-Server aufgebaut werden, so übernimmt der PPP-Dämon die meisten Aufgaben, dadurch wird PPP viel einfacher und sicherer.

Will ein anderer Rechner im TCP/IP-Netzwerk mit einem anderen Rechner kommunizieren, so kennt er zwar dessen IP-Adresse, nicht jedoch (im Falle eines Ethernet) die *Hardware-Adresse*, an die die Datagramme gesandt werden müssen. Der ARP-Mechanismus führt nun genau diese Übersetzung von Netzwerk- und Hardware-Adressen durch. Zu diesem Zweck wird ein spezielles Datagramm, welches die Adresse desjenigen Rechners enthält, dessen Hardware-Adresse gesucht ist, an die Broadcast-Adresse gesandt. Damit 'hören' alle Rechner im Netz diesen Aufruf, und derjenige, dessen Adresse gesucht ist, reagiert darauf, indem er seine Hardware-Adresse zurücksendet. Damit kennt nun der Rechner, der die ARP-Anfrage gestartet hatte, die Hardware-Adresse und kann seine Datagramme weiterleiten.

Ist ein Rechner nun als Server für weitere Rechner konfiguriert, so muß er auch auf alle ARP-Anfragen reagieren, die an diese Klienten gerichtet sind, da sie ja nicht physikalisch im Ethernet-Netzwerk eingebunden sind. Dies erläutert sich am besten an einem praktischen Beispiel: Ein Rechner hat in seinem lokalen Netzwerk einige IP-Adressen, die er für SLIP-Verbindungen über Modem zur Verfügung stellt. Dies seien die Adressen `128.253.154.120-124`. Dieser Rechner ist weiter über eine Ethernet-Karte mit der Hadrware-Adresse `00:00:C0:AD:37:1C` mit dem lokalen Netz verbunden (die Hardware-Adresse kann man mit dem `ifconfig`-Befehl herausfinden). Um diesem Linux-Server nun mitzuteilen, daß er auch die ARP-Anfragen für die SLIP-Adressen beantworten soll, müssen die folgenden Zeilen an die Datei `rc.inet1` angefügt werden:

```
#
# Proxy ARP for those dialin users who will be using this
#           machine as a server:
#
/sbin/arp -s 128.263.154.120 00:00:C0:AD:37:1C pub
/sbin/arp -s 128.263.154.121 00:00:C0:AD:37:1C pub
/sbin/arp -s 128.263.154.122 00:00:C0:AD:37:1C pub
/sbin/arp -s 128.263.154.123 00:00:C0:AD:37:1C pub
/sbin/arp -s 128.263.154.124 00:00:C0:AD:37:1C pub
#
# End proxy arps.
```

Der Parameter `pub` steht für *publish* (öffentlich machen). Dies ist der Befehl für den lokalen Rechner, auch die Anfragen für die angegebene Adresse zu beantworten, obwohl es nicht die eigene ist. Dabei wird mit der angegebenen Hardware-Adresse geantwortet, welches natürlich wiederum die eigene ist.

Selbstverständlich ist es außerdem notwendig, daß auf dem Linux-Server auch Routes zu diesen Adressen konfiguriert sind, die auf die jeweiligen SLIP-Devices zeigen.

Bei der Benutzung von PPP braucht man sich nicht darum kümmern, diese ARP-Tabelle manuell auf dem laufenden zu halten, denn dies wird von `pppd` automatisch durchgeführt, wenn die Option `proxyarp` angegeben wurde. Voraussetzung dabei ist, daß die IP-Adresse von Server und Klient zum selben Netzwerk gehören. Aus diesem Grund muß man auf dem Server beim Start des `pppd` die Netzwerk-Maske in der Kommandozeile mitangeben.

10.4 Gated - der Routing Dämon

Anstelle von *Proxy ARP* hätte man im vorangegangenen Beispiel auch `gated` verwenden können. Dessen Hauptaufgabe liegt aber darin, einen Linux-Rechner als intelligenten *IP Router* in einem Netzwerk zu konfigurieren. `gated` bietet Unterstützung für eine ganze Reihe an Routing-Protokollen: RIP, BGP, EGP, HELLO, OSPF und ein paar mehr. Insbesondere in kleineren Netzen ist *RIP* das verbreitetste. RIP steht dabei für *Routing Information Protocol*. Ein Linux-Rechner, auf dem `gated` für RIP konfiguriert ist, verbreitet in regelmäßigen Abständen seine eigene Routing-Tabelle in einem besonderen Format über die Broadcast-Adresse. Auf diese Weise sind alle Rechner im Netz stets darüber informiert, welche Adressen über diesen Rechner erreicht werden können.

Wenn auf allen Rechnern in einem Netzwerk entweder `gated` oder `routed` läuft, kann *Proxy ARP* durch `gated` ersetzt werden. In einem gemischen Netzwerk, welches sowohl statische als auch dynamische Routes verwendet, sollten sämtliche statischen Routes als `passiv` erklärt werden um sicherzustellen, daß sie von `gated` nicht gelöscht werden, weil er für sie keine wiederholten Gltigkeitsmeldungen bekommt. Statische Routes deklariert man bei der Verwendung von `gated` am besten durch einen `static`-Eintrag in der Datei `/etc/gated.conf`. Dies wird weiter unten beschrieben.

Wie es sich für einen Netzwerk-Dämon gehört, wird `gated` für gewöhnlich in der rc-Datei `rc.inet2` gestartet. Eventuell wird man feststellen, daß bereits ein Dämon namens `routed` läuft. Da `gated` weit flexibler ist als `routed` sollte man dessen Verwendung vorziehen und den `routed`-Prozess nicht weiter verwenden.

10.4.1 Woher bekomme ich gated ?

`gated` kann per Anonymous FTP bezogen werden von:

```
sunsite.unc.edu:/pub/Linux/system/Network/daemons/gated11_bin.tgz
```

10.4.2 Die Installation von gated

Die binäre Paket von `gated` besteht aus drei Programmen und zwei beispielhaften Konfigurationsdateien. Diese sind:

gated

der eigentliche `gated` Dämon.

gdc

die (optionale) Benutzerschnittstelle zu `gated`. Mittels *gdc* kann man den `gated`-Dämon anhalten, neu straten, Statusabfragen durchführen usw.

ripquery

Ein Diagnoseprogramm, mit dem man die bekannten Routes entweder durch eine 'rip query' oder ein 'rip poll' abfragen kann.

Die Konfigurations-Dateien sind:

gated.conf

dies ist die zentrale Konfigurations-Datei für den `gated`-Dämon. Hier kann das gesamte Verhalten von `gated` festgelegt werden, es können die verschiedenen Protokolle aktiviert und deaktiviert werden und die aktiven Protokolle können in ihrem Verhalten beeinflußt werden.

gated.version

eine kleine Text-Datei, die die Versionsnummer des `gated`-Dämons enthält.

Unerfreulicherweise werden die verschiedenen Dateien von der Binär-Distribution nicht an die richtigen Stellen installiert. Da es aber nur wenige sind, kann man das sehr einfach beheben.

Um sie zunächst temporär zu entpacken und dann an die richtigen Stellen zu installieren dienen die folgenden Befehle:

```
% cd /tmp
% gzip -dc .../gated.linux.bin.tgz | tar xvf -
% install -m 500 bin/gated /usr/sbin
% install -m 444 bin/gated.conf bin/gated.version /etc
% install -m 555 bin/ripquery bin/gdc /sbin
% rm -rf /tmp/bin
```

/usr/sbin ist dabei der übliche Platz für Netzwerk-Dämonen. Wer sie auf seinem System in einem anderen Verzeichnis hat, muß das Script natürlich entsprechend abändern. Die mitgelieferte Beispielkonfiguration setzt gated so auf, daß das Verhalten des älteren routed-Dämons emuliert wird. In den meisten Fällen wird dies funktionieren. Die Datei /etc/gated.conf sieht dafür folgendermassen aus:

```
#
#  This configuration emulates routed.  It runs RIP and only sends
#  updates if there are more than one interfaces up and IP forwarding is
#  enabled in the kernel.
#
#       NOTE that RIP *will not* run if UDP checksums are disabled in
#       the kernel.
#
rip yes ;
traceoptions all;
#
```

Gibt es im Netz irgendwelche statischen Routes, sollten diese ebenfalls in diese Datei eingetragen werden. Dies geschieht, indem ein solcher Eintrag hinzugefügt wird:

```
#
static {
37.0.0.0 mask 255.0.0.0 gateway 44.136.8.97 ;
host 44.136.8.100 gateway 44.136.8.97 ;
} ;
#
```

In diesem Beispiel wird eine statische Route zu einem Class-A Netzwerk 37.0.0.0 über den Gateway 44.136.8.97 angelegt sowie eine weitere zu dem Rechner mit der Adresse 44.136.8.100 ebenfalls über den Gateway 44.136.8.97.

Will man auch die Dateien der Online-Hilfe für gated installieren, geht man wie folgt vor:

```
% cd /tmp
% gzip -dc .../gated.linux.man.tgz | tar xvf -
% install -m 444 man/*.8 /usr/man/man8
% install -m 444 man/*.5 /usr/man/man5
% rm -rf /tmp/man
```

Diese man Dateien enthalten sehr präzise und ausführliche Hinweise zur Benutzung von gated. Informationen zur Konfiguration erhält insbesondere das gated-config Manual.

11 Die Konfiguration der Netzwerk-Dämonen

Zur Vervollständigung der Netzwerk-Konfiguration sind außer den beiden rc.inet*-Dateien noch weitere Konfigurationsdateien nötig. Diese kontrollieren das Verhalten des Netzwerk-Codes auf einer etwas höheren Ebene, nämlich dasjenige der Netzwerk-Dämonen, die u.a. in rc.inet2 gestartet werden. Die folgenden Unterabschnitte erläutern die wichtigsten davon.

11.1 /etc/rc.d/rc.inet2 (bzw. zweite Hälfte von rc.net)

Wer den Anweisungen des Textes bis zu diesem Punkt gefolgt ist besitzt nun eine `rc`-Datei, die jedes der vorhandenen Netzwerk-Devices korrekt konfiguriert und alle benötigten Routes zu den erreichbaren Netzwerken deklariert. Nun müssen die eigentlichen, höheren Netzwerk-Programme gestartet werden.

Das wäre genau der richtige Zeitpunkt, den *Network Administrators Guide* von Olaf Kirch zu lesen, denn er ist **das** Buch zu diesem Thema schlechthin. Es gibt alle notwendigen Hinweise, was in diese Datei aufgenommen werden sollte, und, fast noch wichtiger, was man hier **nicht** hineintun sollte. Im Hinblick auf die Systemsicherheit kann man nämlich ganz knapp sagen: Je mehr Netzwerk-Dienste konfiguriert und angeboten werden, desto höher ist das Risiko, ein Sicherheitsloch aufzutun. Also immer nur das konfigurieren, was auch wirklich benötigt wird.

In jedem Fall muß man einiges über die wichtigen Dämonen wissen, das sind Programme, die im Hintergrund ablaufen und viele Dinge automatisch regeln. Die Online-Hilfe mittels `man` gibt hier genauere Information, hier deshalb nur eine kurze Zusammenfassung:

11.1.1 inetd

`inetd` verwaltet sämtliche Anfragen zu Internet-Verbindungen und ähnlichem. Der große Vorteil von `inetd` ist, daß man nicht für alle angebotenen Dienste einen eigenen Serverprozeß laufen lassen muß, auch wenn momentan dieser Dienst gar nicht angefordert ist. Erst wenn eine Anfrage für einen bestimmten Dienst, z.B. `telnet`, eintrifft, sieht `inetd` in der Datei `/etc/services` nach, welches Programm diesen Dienst zur Verfügung stellt. Dieses Programm wird dann gestartet und die Verbindung an es weitergegeben. Man kann sich `inetd` also als eine Art Internet-Hauptserver vorstellen. Einige Standard-Dienste sind übrigens direkt in `inetd` implementiert, dies sind `echo`, `discard` ud `generate`, die für diverse Netzwerk-Tests verwendet werden. `inetd` kann nicht alle Dienste und Server-Programme verwalten, aber doch die am meisten verwendeten. Dienste wie z.B. solche die auf `udp` basieren, oder solche, die das Multiplexing ihrer Verbindungen selber regeln, wie World Wide Web oder MUDs müssen unabhängig von `inetd` gestartet werden. Im Normalfall wird in der Dokumentation solcher Pakete ein Hinweis gegeben, ob der jeweilige Dienst über `inetd` verwaltet werden sollte oder nicht.

11.1.2 syslogd

`syslogd` verwaltet das *System Logging*, also all die vielen Meldungen, die die verschiedenen Dämonen und Kernel-Module bei ihrer Arbeit produzieren. `syslogd` verteilt diese Meldungen gemäß einem Schema, welches in der Datei `/etc/syslogd.conf` festgelegt wird, entweder auf unterschiedliche Protokoll-Dateien oder verwirft sie. So können z.B. bestimmte Meldungen sowohl auf der Konsole ausgegeben als auch mitprotokolliert werden, andere hingegen nur ein einer Datei gespeichert werden.

11.2 Beispiel für eine rc.inet2 Datei

Das folgende Beispiel wurde von Fred van Kempen erstellt. Es startet eine große Zahl an Servern, es sollte also genau studiert und auf den für den eigenen Bedarf angemessenen Umfang reduziert werden. Die geschieht am besten, indem man die nicht benötigten Abschnitte durch ein Kommentarzeichen # deaktiviert, dann können sie zu einem späteren Zeitpunkt leichter wieder implementiert werden, sollte sich das eigene Anforderungsprofil ändern. Diese einzelnen Abschnitte bestehen jeweils aus einem Aufruf des entsprechenden Programms, welcher in einer `if ... fi`-Abfrage eingebettet ist. Diese tut nichts anderes als zu prüfen, ob die angegebene Datei auch existiert und ausführbar ist, um Fehlermeldungen zu vermeiden. Weiterhin wird ein kurzer Hinweis über das gestartete Programm ausgegeben. Hinweise zu den hier aufgeführten Dämonen entnimmt man entweder dem *Network Administrators Guide* oder der Online-Hilfe mittels `man`.

```
#! /bin/sh
#
```

```
# rc.inet2      This shell script boots up the entire INET system.
#               Note, that when this script is used to also fire
#               up any important remote NFS disks (like the /usr
#               distribution), care must be taken to actually
#               have all the needed binaries online _now_ ...
#
# Version:      @(#)/etc/rc.d/rc.inet2  2.18    05/27/93
#
# Author:       Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#

# Konstanten
NET="/usr/sbin"
IN_SERV="lpd"
LPSPOOL="/var/spool/lpd"

# Jetzt koennen auch NFS-Dateisysteme gemounted werden, das kann
# z.B. auch der gesamte /usr-Verzeichnisbaum sein
echo -e "\nMounting remote file systems ..."
/bin/mount -t nfs -v

echo -e "\nStarting Network daemons ..."
# Als erste wird der SYSLOG Daemon gestartet.  Dies muss der
# erste Server sein, er ist IN JEDEM FALLE notwendig!
echo -n "INET: "
if [ -f ${NET}/syslogd ]
then
echo -n "syslogd "
${NET}/syslogd
fi

# Starte den SUN RPC Portmapper.
if [ -f ${NET}/rpc.portmap ]
then
echo -n "portmap "
${NET}/rpc.portmap
fi

# Starte den INET SuperServer
# Auch dieser ist IMMER NOTWENDIG!
if [ -f ${NET}/inetd ]
then
echo -n "inetd "
${NET}/inetd
else
echo "no INETD found.  INET cancelled!"
exit 1
fi

# Starte den NAMED/BIND Nameserver.
# HINWEIS: named wird fuer endbenutzer-Rechner meist nicht benoetigt.
#if [ ! -f ${NET}/named ]
```

```
#then
#        echo -n "named "
#        ${NET}/named
#fi

# Starte den ROUTEd Server.
# HINWEIS: routed ist veraltet, stattdessen wird gated benutzt.
#if [ -f ${NET}/routed ]
#then
#        echo -n "routed "
#        ${NET}/routed -q #-g -s
#fi

# Starte den GATEd Server.
if [ -f ${NET}/gated ]
then
echo -n "gated "
${NET}/gated
fi

# Starte den RWHO Server.
if [ -f ${NET}/rwhod ]
then
echo -n "rwhod "
${NET}/rwhod -t -s
fi

# Starte den U-MAIL SMTP Server.
if [ -f XXX/usr/lib/umail/umail ]
then
echo -n "umail "
/usr/lib/umail/umail -d7 -bd </dev/null >/dev/null 2>&1 &
fi

# Starte die verschiedenen INET Server.
for server in ${IN_SERV}
do
if [ -f ${NET}/${server} ]
then
                echo -n "${server} "
                ${NET}/${server}
fi
done

# Starte die diversen SUN RPC Server.
if [ -f ${NET}/rpc.portmap ]
then
if [ -f ${NET}/rpc.ugidd ]
then
                echo -n "ugidd "
                ${NET}/rpc.ugidd -d
fi
```

```
if [ -f ${NET}/rpc.mountd ]
then
                echo -n "mountd "
                ${NET}/rpc.mountd
fi
if [ -f ${NET}/rpc.nfsd ]
then
                echo -n "nfsd "
                ${NET}/rpc.nfsd
fi

# Starte den/die PC-NFS Daemon(en).
if [ -f ${NET}/rpc.pcnfsd ]
then
                echo -n "pcnfsd "
                ${NET}/rpc.pcnfsd ${LPSPOOL}
fi
if [ -f ${NET}/rpc.bwnfsd ]
then
                echo -n "bwnfsd "
                ${NET}/rpc.bwnfsd ${LPSPOOL}
fi

fi
echo network daemons started.
# Fertig!
```

11.3 Weitere notwendige Netzwerk-Konfigurationsdateien

Soll es anderen Personen ermöglicht werden, Verbindung mit dem eigenen Rechner aufzunehmen, so sind einige weitere Konfigurationsdateien notwendig. Wer ein Linux-System aus einer zusammengestellten Distribution installiert hat, wird diese Dateien vermutlich schon an den entsprechenden Stellen besitzen. Dann genügt es, diese auf Korrektheit zu überprüfen. Ansonsten sind hier Standardbeispiele aufgeführt.

11.3.1 Beispiel für eine /etc/inetd.conf Datei

In `/etc/rc.d/rc.inet2` werden der `inetd`- und der `syslogd`-Dämon sowie diverse `rpc` Server gestartet. Die von `inetd` verwalteten Dienste müssen in der Datei `/etc/inetd.conf` konfiguriert werden. Das folgende Beispiel zeigt eine solche einfache Konfiguration:

```
#
# The internal services.
#
# Authors:      Original taken from BSD UNIX 4.3/TAHOE.
#               Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
echo    stream tcp nowait root  internal
echo    dgram  udp wait   root  internal
discard stream tcp nowait root  internal
discard dgram  udp wait   root  internal
daytime stream tcp nowait root  internal
```

```
daytime dgram  udp wait   root  internal
chargen stream tcp nowait root  internal
chargen dgram  udp wait   root  internal
#
# Standard services.
#
ftp     stream tcp nowait root  /usr/sbin/tcpd in.ftpd ftpd
telnet  stream tcp nowait root  /usr/sbin/tcpd in.telnetd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream tcp nowait root  /usr/sbin/tcpd in.rshd
login   stream tcp nowait root  /usr/sbin/tcpd in.rlogind
exec    stream tcp nowait root  /usr/sbin/tcpd in.rexecd
talk    dgram  udp wait   root  /usr/sbin/tcpd in.talkd
ntalk   dgram  udp wait   root  /usr/sbin/tcpd in.talkd
#
# Status and Information services.
#
finger  stream tcp nowait root  /usr/sbin/tcpd in.fingerd
systat  stream tcp nowait guest /usr/sbin/tcpd /usr/bin/ps -auwwx
netstat stream tcp nowait guest /usr/sbin/tcpd /bin/netstat
#
# End of inetd.conf.
```

Die Online-Hilfe zu `inetd` beschreibt ausführlich, was jedes dieser Felder darstellt. Kurz gesagt wird für jeden der in der ersten Spalte aufgeführten Dienste angegeben, welches Programm gestartet werden soll, wenn dieser Dienst angefordert wird. Diejenigen Einträge, bei denen im letzten Feld `internal` steht, werden von `inetd` selbst zur Verfügung gestellt.

Die Übersetzung vom Namen eines Dienstes (aus der ersten Zeile) und der *Socket Nummer*, über die dieser Dienst erreichbar ist, wird in der Datei `/etc/services` festgelegt.

11.3.2 Beispiel für die Datei /etc/services

Die Datei `/etc/services` ist eine einfache Tabelle der Namen der Internet-Dienste, ihrer Socket-Nummern und des verwendeten Protokolls. Diese Tabelle wird von einigen Programmen verwendet, darunter `inetd`, `telnet` und `tcpdump`. Die Zuordnung von Namen dient dabei übrigens nur dazu, daß man sich die Dienste leichter merken kann. Dem Rechner ist es egal, ob er mit Namen oder Nummern arbeitet.

Eine typische `/etc/services` Datei sieht so aus:

```
#
# /etc/services - database of service name, socket number
#                 and protocol.
#
# Original Author:
#     Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
tcpmux     1/tcp
echo       7/tcp
echo       7/udp
discard    9/tcp   sink null
```

```
discard    9/udp   sink null
systat     11/tcp  users
daytime    13/tcp
daytime    13/udp
netstat    15/tcp
chargen    19/tcp  ttytst source
chargen    19/udp  ttytst source
ftp-data   20/tcp
ftp        21/tcp
telnet     23/tcp
smtp       25/tcp  mail
time       37/tcp  timserver
time       37/udp  timserver
name       42/udp  nameserver
whois      43/tcp  nicname    # usually to sri-nic
domain     53/tcp
domain     53/udp
finger     79/tcp
link       87/tcp  ttylink
hostnames  101/tcp hostname   # usually to sri-nic
sunrpc     111/tcp
sunrpc     111/tcp portmapper # RPC 4.0 portmapper TCP
sunrpc     111/udp
sunrpc     111/udp portmapper # RPC 4.0 portmapper UDP
auth       113/tcp authentication
nntp       119/tcp usenet     # Network News Transfer
ntp        123/tcp            # Network Time Protocol
ntp        123/udp            # Network Time Protocol
snmp       161/udp
snmp-trap  162/udp
exec       512/tcp            # BSD rexecd(8)
biff       512/udp comsat
login      513/tcp            # BSD rlogind(8)
who        513/udp whod       # BSD rwhod(8)
shell      514/tcp cmd        # BSD rshd(8)
syslog     514/udp            # BSD syslogd(8)
printer    515/tcp spooler    # BSD lpd(8)
talk       517/udp            # BSD talkd(8)
ntalk      518/udp            # SunOS talkd(8)
route      520/udp routed     # 521/udp too
timed      525/udp timeserver
mount      635/udp            # NFS Mount Service
pcnfs      640/udp            # PC-NFS DOS Authentication
bwnfs      650/udp            # BW-NFS DOS Authentication
listen     1025/tcp listener  # RFS remote_file_sharing
ingreslock 1524/tcp           # ingres lock server
nfs        2049/udp           # NFS File Service
irc        6667/tcp           # Internet Relay Chat
# End of services.
```

Hier zeigt z.B. der Eintrag bei `telnet`, daß dieser Dienst Socket 23 und das `tcp`-Protokoll verwendet, und daß der Domain Name Service `domain` Socket 52 benutzt und als Protokoll sowohl `tcp` als auch `udp`. Wichtig ist in jedem

Fall, das für jeden Eintrag in `/etc/inetd.conf` ein entsprechender in `/etc/services` vorhanden ist.

11.3.3 Beispiel für die Datei /etc/protocols

Die Datei `/etc/protocols` enthält eine Tabelle der Protokollnamen und ihrer Protokollnummern. Da es nur recht wenig verwendete Protokolle gibt, ist diese Datei recht kurz und einfach:

```
#
# /etc/protocols - database of protocols.
#
# Original Author:
#   Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
ip    0   IP   # internet protocol
icmp 1    ICMP # internet control message protocol
igmp 2    IGMP # internet group multicast protocol
ggp  3    GGP  # gateway-gateway protocol
tcp  6    TCP  # transmission control protocol
pup  12   PUP  # PARC universal packet protocol
udp  17   UDP  # user datagram protocol
idp  22   IDP
raw  255 RAW
#
# End of protocols.
```

11.4 Name Resolution

Als *Name Resolution* bezeichnet man den Prozeß, einen Rechnernamen (z.B. `tsx-11.mit.edu`) in die entsprechende IP-Adresse in punktierter Dezimalschreibweise zu übersetzen, die die Netzwerk-Software benötigt. Hierfür gibt es zwei prinzipielle Möglichkeiten, eine einfache und eine etwas komplexere.

11.4.1 /etc/hosts

`/etc/hosts` enthält eine Liste von IP-Adressen und der Rechnernamen, die dieser Adresse entsprechen. Auf diese Weise können die dort angegebenen Rechner durch ihre Namen angesprochen werden, anstatt die (Nummern)-Adresse zu verwenden. Die Benutzung eins Nameservers (siehe Abschnitt 'named') erlaubt, dasselbe automatisch durchzuführen (mittels `named` kann man auch seinen eigenen Rechner als Nameserver konfigurieren). `/etc/hosts` muß mindestens einen Eintrag für die Adresse `127.0.0.1` mit dem Namen `localhost` enthalten, wer kein Loopback benutzt, muß außerdem die eigene Adresse mit vollem Rechnernamen (also Hostname und Domainname, `loomer.vpizza.com`) hinzufügen. Ebenfalls empfehlenswert ist es, Einträge für die verwendeten Gateways, Nameserver und evtl. lokale Adressen anzulegen.

Wenn zum Beispiel der Rechner `loomer.vpizza.com` die IP-Adresse `128.253.154.32` besitzt, sähe ein entsprechender Eintrag in `/etc/hosts` folgendermassen aus:

```
# /etc/hosts
# List of hostnames and their ip addresses
127.0.0.1               localhost
128.253.154.32          loomer.vpizza.com loomer
# end of hosts
```

Diese Datei muß in jedem Fall editiert und den eigenen Bedürfnissen angepaßt werden. Wer Loopback verwendet, benötigt darin aber lediglich einen einzigen Eintrag: `127.0.0.1`, wobei sowohl `localhost` als auch der Rechnername eingetragen sind.

In der zweiten Zeile im obigen Beispiel sind übrigens zwei Namen für die Adresse `128.253.154.32` eingetragen: `loomer.vpizza.com` und nur `loomer`. Der erste Eintrag ist der volle Name des Rechners, den man als "Fully Qualified Domain Name" (FQDN) bezeichnet, der zweite Eintrag ist ein alias dafür. Dies erlaubt es, den Rechner unter diesem Namen anzusprechen und z.B. nur `rlogin loomer` einzugeben, anstatt den vollen Namen zu verwenden. In jedem Fall sollte der FQDN der **erste** Eintrag nach der IP-Adresse sein.

11.4.2 Named - brauche ich das ?

`named` ist der Nameserver Dämon in vielen Unix-ähnlichen Betriebssystemen. Er erlaubt es dem Rechner, die Adressen-Namens-Zuordnung nicht nur für sich selbst sondern auch für anderer Rechner im Netzwerk durchzuführen. D.h. wenn ein anderer Rechner z.B. die Adresse des Rechners `goober.norelco.com` sucht und sich dieser Name in der Datenbasis von `named` befindet, dann kann der eigene Rechner dem anderen die gesuchte IP-Adresse mitteilen.

Unter früheren Implementationen von TCP/IP unter Linux war es erforderlich, einen `named`-Server zu installieren, wenn man Aliase für Rechnernamen verwenden wollte, selbst wenn es für den eigenen Rechner war. Das Problem dabei ist, daß `named` relativ kompliziert zu konfigurieren ist. Hierfür gab es ein Hilfsprogramm mit dem Namen `hostcvt.build`. Dieses konvertierte die Datei `/etc/hosts` in die vielen Einzeldateien, die die Datenbasis für `named` darstellen. Doch selbst mit dieser Vereinfachung verursacht `named` einiges an zusätzlicher CPU- und Netzwerk-Belastung.

Aus diesem Grund lohnt sich der Einsatz von `named` nur in folgenden Fällen:

- Es soll ein ganzes Netzwerk von Rechnern zusammengestellt werden, und es steht kein externer Nameserver zur Verfügung.
- Der Netzwerk-Administrator möchte den Linux-Rechner als Nameserver verwenden.
- Es besteht nur eine langsame SLIP-Verbindung zum Netz. Dann lohnt es sich, einen kleinen Nameserver zu konfigurieren, der nur einen Cache anlegt, damit nicht für jede Adresse eine (langsame) Anfrage über die serielle Leitung gemacht werden muß. Wer allerdings bereits absehen kann, daß er nur mit einer kleinen Zahl von Rechnern Verbindung aufnehmen will, deren IP-Adressen er kennt, für den empfiehlt sich die Übernahme dieser Nummern/Adressen-Paare in die Datei `/etc/hosts`. Ein Nameserver ist dann nicht nötig.
- Der Nameserver soll zu Lernzwecken oder 'just for fun' installiert werden.

Generell kann man zusammenfassen: **named wird nicht benötigt**. Er kann deshalb in der Datei `rc.inet2` auskommentiert werden. Aliasing von Rechnernamen ist unter den aktuellen Implementationen auch über die Datei `/etc/hosts` möglich, es müssen lediglich die gewünschten zusätzlichen Namen hinter dem vollen Namen (FQDN) angegeben werden. Solange man Zugriff auf einen Nameserver hat (dessen Adresse kann man vom Netzwerk-Administrator erfahren) gibt es keinen Grund, selber `named` zu benutzen.

Wer Loopback verwendet kann trotzdem einen Nameserver mittels `named` konfigurieren, als Adresse für den Nameserver muß man dann `127.0.0.1` verwenden. Allerdings ist das recht bizarr, denn der eigene Rechner ist ja der einzige, mit dem man kommunizieren kann, sodaß eine Anfrage an `named` nie stattfinden wird.

11.4.3 /etc/networks

Die Datei `/etc/networks` enthält eine Liste der Namen und Adressen der bekannten Netzwerke, also zumindest diejenige des lokalen Netzwerkes. `route` benutzt diese Datei, wenn das Netzwerk mit dem Namen anstatt seiner Adresse angegeben wird, um aus diesem Namen eine gültige Adresse zu erfahren.

Es empfiehlt sich, für jedes Netzwerk, zu dem explizit eine Route angelegt werden soll, einen solchen Eintrag in `/etc/networks` anzufügen. Ist ein Netzwerk nicht in dieser Datei aufgeführt, so **muß** der Parameter `-net` mit `route`-Befehl benutzt werden um anzuzeigen, daß es sich bei der Adresse um ein Netzwerk und nicht einen einzelnen Rechner handelt.

Das Format der Datei ist ganz ähnlich zu demjenigen von `/etc/hosts`, eine typische Datei sieht folgendermassen aus:

```
#
# /etc/networks: list all networks that you wish to add route commands
#                 for in here
#
default         0.0.0.0         # default route    - recommended
loopnet         127.0.0.0       # loopback network - recommended
mynet           128.253.154.0   # Example network CHANGE to YOURS
#
# end of networks
```

11.4.4 /etc/host.conf

Das System stellt einige Bibliotheks-Routinen zur Verfügung, die man die *Resolver Library* nennt. Die Datei `/etc/host.conf` legt fest, wie das System bei der Übersetzung von Rechnernamen in gültige IP-Adressen vorgeht. Mindestens zwei Einträge sollten vorhanden sein:

```
order hosts,bind
multi on
```

Diese beiden Zeilen weisen die Bibliotheksroutinen an, die Rechneradressen zunächst in der Datei `/etc/hosts` zu suchen und dann, falls die gesuchte Adresse dort nicht gefunden wurde, einen Nameserver zu fragen. Der Eintrag `multi` erlaubt es dabei, in der Datei `/etc/hosts` mehrere Adresseinträge für einen bestimmten Rechnernamen zu haben.

Diese Datei entstammt der Implementation der `resolv+` bind Library unter Linux. Weitere Informationen dazu enthält die Online-Hilfe zu `resolv+(8)`, allerdings ist sie nicht bei allen Distributionen standardmäßig enthalten. In diesem Fall kann man sie sich hier besorgen:

`sunsite.doc.ic.ac.uk:/computing/comms/tcpip/nameserver/resolv+/resolv+2.1.1.tar.Z`

11.4.5 /etc/resolv.conf

`/etc/resolv.conf` konfiguriert den Name-Resolver des Systems, der die Rechnernamen in IP-Adressen übersetzt. Es gibt zwei unterschiedliche Arten von Einträgen in dieser Datei: Die Adressen des oder der Nameserver(s) sowie die Domain-Adresse. Wer auf dem lokalen Rechner selber `named` als Nameserver betreibt, der muß als Nameserveradresse die Loopbackadresse `127.0.0.1` angeben.

Der *Domain Name* besteht aus dem vollen Rechnernamen (FQDN) ohne den eigentlichen Namensanteil, also für das Beispiel von vorhin (`loomer.vpizza.com`) wäre der Domain Name `vpizza.com`.

Für einen Rechner mit dem (vollen) Namen `goober.norelco.com`, dessen Nameserver die Adresse `128.253.154.5` besitzt, sähe die Datei `/etc/resolv.conf` dann folgendermaßen aus:

```
domain norelco.com
nameserver 128.253.154.5
```

Es können auch mehrere Nameserver eingetragen werden, dann muß jede Adresse in eine eigene Zeile geschrieben werden, jeweils mit dem Schlüsselwort `nameserver` am Zeilenanfang.

Wer lediglich im Loopbackbetrieb arbeitet braucht wie bereits erwähnt keinen Nameserver.

11.4.6 Die Festlegung des Rechnernamens - /etc/HOSTNAME

Nachdem nun alles andere konfiguriert ist fehlt nur noch eine Kleinigkeit: der eigene Rechner muß seinen Namen mitgeteilt bekommen. Denn Programme wie `sendmail` müssen schließlich wissen, für welchen Rechner sie die mail in Empfang nehmen sollen, und sie müssen sich ja auch anderen Rechnern gegenüber identifizieren.

Diese Konfiguration des Namens geschieht über zwei unterschiedliche Programme, `hostname` und `domainname`. Leider wird ihre Anwendung manchmal durcheinandergebracht.

Wer eine Version der `net-tools` älter als `1.1.38` benutzt, kann in seine `/etc/rc`-Datei folgenden Befehl aufnehmen:

```
/bin/hostname -S
```

`hostname` wird dann die Datei `/etc/HOSTNAME` lesen, in der der volle Rechnername (FQDN), also Rechnername und Domain-Name, stehen muß. Dieser Name wird dann von `hostname` in Host- und Domain-Name aufgespalten und zur Konfiguration verwendet.

Die Versionen der `net-tools-1.1.38` später sollten in ihrer Datei `/etc/rc.d/rc.inet1` folgende Zeilen am Ende einfügen:

```
/bin/hostname goober.norelco.com
```

oder, falls man en Upgrade einer früheren Version durchgeführt hat oder einfach den Rechnernamen lieber in `/etc/HOSTNAME` stehen hat:

```
/bin/hostname -F /etc/HOSTNAME
```

Dadurch erreicht man dasselbe Verhalten wie bei der früheren Version.

Der Befehl `/bin/domainname` dient **nicht** der Festlegung des DNS Domain Namens, sondern er bestimmt den **N.I.S.** Domain Namen. Dieser wird aber nur benötigt, wenn auch *NIS* verwendet werden soll, das später noch kurz beschrieben wird.

11.5 Weitere Dateien

Natürlich gibt es im Verzeichnis noch eine ganze Menge an Dateien, mit denen man sich früher oder später wird befassen müssen. Hier soll aber nur eine Minimalkonfiguration beschrieben werden, um den Rechner netztauglich zu machen. Für weitere Informationen sei nochmals auf den *Network Administration Guide* von Olaf Kirch verwiesen. Er steigt dort ein, wo dieses *HOWTO* aufhört.

Nachdem nun alles notwendige konfiguriert ist steht einem Reboot des neuen Kernels nichts mehr entgegen, um nach Herzenslust im Netz zu surfen. In jedem Fall empfiehlt es sich aber, eine Version des alten Kernels aufzubewahren, entweder in Form einer Boot-Diskette oder als zusätzlicher Menüpunkt für LILO. Der sicherste Weg ist in jedem Fall eine Rettungsdiskette mit einem eigenen lauffähigen System, nur für den Fall, daß etwas schiefgeht. Inzwischen gibt es einige solcher Rettungssysteme, z.B. HJLu's 'single disk boot disk', die boot/root Disketten einer beliebigen Distribution oder ein mit *yard* selber zusammengestelltes System.

12 Fortgeschrittene Konfiguration

Die oben beschriebene Konfiguration hat gezeigt wie eine typische Linux-Workstation für den normalen Endbenutzer-Betrieb konfiguriert wird. Einige werden aber besondere Wünsche und Bedürfnisse haben, die eine etwas fortgeschrittenere Konfiguration benötigt. Einige der gängigsten sollen in den kommenden Abschnitten beschrieben werden.

Die Details zu AX.25, Ottawa PI und den generischen SCC Treibern wurden aus diesem Text herausgenommen, ihnen widmet sich ein eigenes HOWTO, das *HAM-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/HAM-HOWTO.html`).

12.1 PPP - Point to Point Protocol

Das *Point to Point Protocol* ist ein modernes und effizientes Protokoll das mehrere verschiedene Protokolle zur Verbindung über serielle Leitungen in sich vereint. Es wird von vielen anstelle von SLIP verwendet, dem es erweiterte Funktionalität, Fehlererkennung und Sicherheitsoptionen vorausha. Es behebt einige SLIP-typische Nachteile und kann sowohl über synchrone wie auch über asynchrone Verbindungen benutzt werden.

Einer der großen Vorteile von PPP ist die Tatsache, daß die Übermittelung von solchen Daten wie der IP-Adresse im Protokoll implementiert ist, und von dieser Möglichkeit machen die meisten Server auch Gebrauch. Dabei sendet der Klient zunächst ein speziell formatiertes Datagramm an den Server und bekommt daraufhin die gewünschte Information - in diesem Falle z.B. die IP-Adresse - zurück, ebenfalls in einer genau festgelegten Form. Dies erleichtern natürlich deutlich den automatischen Aufbau einer dynamischen Verbindung gegenüber SLIP, bei dem man - außerhalb des Protokolls - auf eine 'schlaue' Software wie `dip` zurückgreifen muß, um diese Aufgabe zu erfüllen.

Die Autoren der Linux-Portierung von PPP sind Michael Callahan, (`callahan@maths.ox.ac.uk>`) und Al Longyear, (`longyear@netcom.com>`), Linux-PPP basiert auf Paul Mackerras's free PPP für BSD-ähnliche Systeme. Der Großteil der hier gegebenen Informationen entstammt der Dokumentation der PPP-Software. Sie ist sehr vollständig und geht über das hier gesagte weit hinaus.

12.1.1 Warum PPP statt SLIP ?

Es gibt mehrere Gründe, warum man PPP anstelle von SLIP verwenden sollte, wenn man die Wahl hat:

Der Internet Provider bietet nur PPP an.

In diesem Falle hat mein natürlich keine Wahl sondern muß PPP verwenden. Dies ist aber sicherlich kein Nachteil ;-)

Schlechte/verrauschte serielle Verbindung.

Im Gegensatz zu SLIP bietet PPP eine Fehlerüberprüfung für jedes einzelne Datenpaket. Deshalb muß unter SLIP die Überprüfung der Daten zwischen Sender und Empfänger eines Datenpaketes erfolgen während unter PPP dies zwischen dem eigenen Rechner und dem PPP-Server erfolgt. Insbesondere wenn zwischen dem PPP/SLIP-Server und dem eigentlichen Sender/Empfänger noch weitere Router im Netz zwischengeschaltet sind erlaubt PPP dadurch eine schnellere Reaktion auf fehlerhafte Datagramme.

Es werden spezielle Möglichkeiten von PPP benötigt.

PPP bietet einige Dinge, die mit SLIP nicht möglich sind. So können z.B. nicht nur IP-Datagramme über die serielle Leitung übermittelt werden sondern auch andere wie DECNET, oder AppleTalk.

12.1.2 Bezugsquellen für die PPP Software

```
sunsite.unc.edu:/pub/Linux/system/Networking/serial/ppp-2.1.2d.tar.gz
sunsite.unc.edu:/pub/Linux/system/Networking/serial/ppp-2.2.0f.tar.gz
```

Diese Dateien enthalten Kernel Quelltexte sowie Quellen und das ausführbare Programm von `pppd`. Wer noch einen Kernel 1.2.x verwendet benötigt die Version 2.1.2d, die andere Version 2.2.0f ist für die neuen 2.0.x-Kernels gedacht.

12.1.3 Installation und Konfiguration der PPP Software

Das PPP-HOWTO beschreibt ausführlich alle Schritte die notwendig sind um die PPP-Software richtig zu installieren, den Rechner über eine PPP-Verbindung mit dem Internet zu verbinden oder selber einen PPP-Server aufzubauen. Auch das PPP-HOWTO gibt es in einer deutschen Übersetzung.

12.2 Linux als SLIP-Server

Es gibt drei verschiedene Möglichkeiten, wie man einen Linux-Rechner als SLIP-Server konfigurieren kann und es so anderen ermöglicht sich über Modem einzuwählen und Netzwerk-Dienste in Anspruch zu nehmen. Die erste, über `sliplogin`, ist wohl am ehesten zu empfehlen, da sie sehr einsichtig und leicht zu konfigurieren ist.

12.2.1 Slip Server mittels sliplogin

`sliplogin` kann anstelle der normalen Login-Shell dazu verwendet werden, die Terminalverbindung für SLIP-Benutzer in den benötigten SLIP-Modus umzuschalten. Dabei kann der Linux-Rechner sowohl als *statischer* Server arbeiten, der den Nutzern, basierend auf ihrer ID, jedesmal dieselbe IP-Adresse zuweist, oder auch als *dynamischer* Server, wobei eine gerade freie Adresse aus einem vorgegebenen Adressbereich vergeben wird.

Der SLIP-Benutzer führt dann zunächst einen ganz normalen Login-Prozeß durch, d.h. er meldet sich mit seinem Benutzernamen an und gibt sein Paßwort ein. Normalerweise würde jetzt die Login-Shell gestartet, die den Benutzer durch ihren Prompt zur Eingabe von Befehlen auffordert. Stattdessen wird für den SLIP-Benutzer nun `sliplogin` ausgeführt. Dieses Programm sucht in seiner Konfigurationsdatei (`/etc/slip.hosts`) nach einem Eintrag, der dem Login-Namen des Benutzers entspricht. Wird dieser gefunden so wird die serielle Verbindung für 8-bit konfiguriert und der Modus durch einen entsprechenden `ioctl` auf SLIP umgesetzt. Als letzten Schritt der Konfiguration führt `sliplogin` dann ein Shellscript aus, welches die SLIP-Schnittstelle mit den entsprechenden Werten für IP-Adresse, Netzwerk-Maske und den dazugehörenden Routing-befehlen konfiguriert. Dieses Script heißt normalerweise `/etc/slip.login`, aber ähnlich wie bei `getty` können benutzerspezifische Scripts mit dem Namen `/etc/slip.login.loginname` angelegt werden, um für jeden Benutzer ein eigens Konfigurationsscript zu haben. Dieses wird dann anstelle des Standardscripts ausgeführt.

Damit `sliplogin` richtig arbeitet müssen drei oder vier Dateien die richtigen Einträge enthalten. Dies sind:

- `/etc/passwd`, zur Verwaltung der Accounts.
- `/etc/slip.hosts`, hierin stehen Informationen für jeden registrierten SLIP-Benutzer.
- `/etc/slip.login`, Konfiguration des Routing für jeden Benutzer.
- `/etc/slip.tty`, diese Datei wird benötigt, wenn der Linux-Rechner als *dynamischer SLIP-Server* arbeiten soll, sie enthält eine Liste der möglichen zu vergebenden Adressen.
- `/etc/slip.logout`, dies sind die Befehle, die zur Beendigung einer SLIP-Verbindung auf der Server-Seite notwendig sind, wenn der Benutzer die Verbindung beendet.

Woher bekomme ich sliplogin ? Die aktuelle Version ist

```
sunsite.unc.edu:/pub/Linux/system/Network/serial/sliplogin-2.0.1.tar.gz
```

Die TAR-Datei enthält die Quellen, vorkompilierte Binärdateien und die Online-Hilfetexte für `man`.

Um sicherzustellen, daß nur autorisierte Benutzer das Programm `sliplogin` ausführen können empfiehlt es sich, dafür eine eigene Gruppe anzulegen, indem in der Datei `/etc/group` ein entsprechender Eintrag eingefügt wird, der ungefähr so aussehen sollte:

```
 ..
SLIP::13:radio,fred
 ..
```

Bei der Installation des `sliplogin` Paketes wird das `Makefile` dann automatisch die Gruppenzugehörigkeit von `sliplogin` auf `SLIP` setzen und die Permissions so einstellen, daß nur Benutzer dieses Programm starten können, die der Gruppe `SLIP` zugehören, im obigen Beispiel als die Benutzer fred und radio.

Die Installation (Programme in `/sbin`, Online-Hilfetexte in Sektion 8) ist Standard:

```
% cd /usr/src
% gzip -dc .../sliplogin-2.0.tar.gz | tar xvf -
   {\(<\)}..editieren des Makefile je nachdem ob Shadow
     Passwoerter verwendet werden oder nicht...>
% cd sliplogin
% make install
```

Damit werden die mitgelieferten Binärdateien installiert. Wer sie lieber selbst kompiliert, muß vor dem `make install` noch ein `make clean` ausführen.

Weitere Informationen enthält die README-Datei.

/etc/passwd für einen SLIP-Server Für die Login-Namen der SLIP-Benutzer, die in `/etc/passwd` festgelegt werden, gibt es eine Konvention, die meist eingehalten wird: Der Name setzt sich zusammen aus dem Namen des einwählenden Rechners, dem ein großes 'S' vorangestellt wird. Der Eintrag für einen Rechner mit dem Namen `radio` sähe dann also folgendermaßen aus:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Hinweis: Als Heimatverzeichnis für diesen Benutzer ist `/tmp` eingetragen, er hat also kein eigenes Verzeichnis. Dieses ist aber für SLIP-Nutzer auch nicht nötig, da anstelle der normalen Login-Shell `sliplogin` ausgeführt wird und es so nie zu einem Dialogbetrieb kommen kann.

/etc/slip.hosts Wenn `sliplogin` bei einem erfolgreichen Einwählversuch gestartet wurde so durchsucht es die Datei `/etc/slip.hosts` nach einem Eintrag der dem Login-Namen des Anrufers entspricht, um die Einzelheiten der Konfiguration für diesen Rechner zu erfahren. Hier können also für statische Accounts IP-Adresse und Netz-Maske festgelegt werden, sowie einige weitere Eigenschaften der SLIP-Verbindung. Das folgende Beispiel zeigt zwei typische Einträge in der Datei `/etc/slip.hosts`:

```
#
Sradio    44.136.8.99    44.136.8.100   0xffffff00   normal
Salbert   44.136.8.99    DYNAMIC        0xffffff00   compressed
#
```

Es treten der Reihe nach folgende Einträge auf:

1. Der Login-Name des Anrufers.

2. Die IP-Adresse des *Servers*, also die eigene Adresse.
3. Die IP-Adresse, die dem *einwählenden Rechner* zugewiesen wird. Enthält dieses Feld den Eintrag `DYNAMIC` dann wird die IP-Adresse entsprechend den Informationen in der Datei `/etc/slip.tty` vergeben, dies wird später behandelt. **Wichtig:** Das funktioniert erst ab Version 1.3 von `sliplogin`!
4. Die zuzuweisende Netz-Maske in hexadezimaler Schreibweise, für ein Klasse-C Netzwerk also z.B. `0xffffff00=255.255.255.0`.
5. Optionale Parameter, z.B. um Kompression ein- oder auszuschalten.

Hinweis: Im 2. und 3. Feld können sowohl IP-Adressen in der üblichen punktierten Dezimalschreibweise (wie im Beispiel) als auch die Rechnernamen verwendet werden. Voraussetzung ist dann aber, daß diese Namen auf dem eigenen Rechner auch aufgelöst werden können (d.h. entsprechende Einträge in `/etc/hosts` vorhanden sind), da dem anderen Rechner sonst keine gültige IP-Adresse zugewiesen werden kann und der Login-Versuch fehlschlägt. Zur richtigen Konfiguration dieser Namensauflösung siehe das Kapitel ‘Name Resolution’.

Häufig verwendete Einträge für die optionalen Einstellungen sind:

normal

verwendet das ‘normale’, unkomprimierte SLIP.

compressed

schaltet die van Jacobsen Header-Komprimierung ein (cSLIP).

Diese beiden Einträge schließen sich gegenseitig aus, es darf also nur eine davon eingetragen werden. Näheres dazu steht in den entsprechenden Abschnitten der `man`-Online-Hilfe.

/etc/slip.login Wenn `sliplogin` in der Datei `/etc/slip.hosts` einen passenden Eintrag für den einwählenden Rechner gefunden hat versucht es als Nächstes, die Datei `/etc/slip.login` auszuführen, um die endgültige Konfiguration des SLIP Device mit IP-Adresse und Netz-Maske durchzuführen. Das `sliplogin`-Paket enthält ein Beispiel für diese Datei:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# Generische Login-Datei fuer eine SLIP-Verbindung.  Sie wird von
# sliplogin mit folgenden Parametern aufgerufen:
#     $1       $2        $3       $4         $5        $6    $7-n
#   SLIPunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig $1 $4 pointopoint $5 mtu 1500 -trailers up
/sbin/route add $5
arp -s $5 <hw_addr> pub
exit 0
#
```

Dieses Script macht also nichts anderes als die `ifconfig` und `route` Befehle mit den übergebenen Parametern auszuführen um das Device richtig zu konfigurieren und eine Route für den neu hinzugekommenen Rechner über das SLIP-Device anzulegen.

Außerdem wird noch *Proxy ARP* verwendet um den anderen Rechnern im lokalen Ethernet des Servers mitzuteilen daß der neue Rechner unter derselben Hardware-Adresse wie der Server erreicht werden kann. Hierzu muß <hw_addr>

im obigen Beispiel durch die Hardware-Adresse der Ethernet-Karte des Servers ersetzt werden. Diese kann durch ein `ifconfig eth0` angezeigt werden.

Wer keine Ethernet-Karte in seinem Rechner hat sollte die gesamte Zeile auskommentieren.

/etc/slip.logout Nach Beendigung der Verbindung muß die serielle Schnittstelle wieder in ihren Normalzustand zurückgesetzt werden, damit der nächste Anrufer wieder einen definierten Zustand vorfindet und ein Login möglich ist. Dies wird in der Script-Datei `/etc/slip.logout` durchgeführt:

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
/sbin/route del $5
arp -d $5
exit 0
#
```

Es wird lediglich das SLIP-Interface als 'down' konfiguriert und die angelegte statische Route wieder aus der Routing-Tabelle gelöscht. Besitzer von Ethernet-Karten müssen weiterhin die *Proxy ARPs* für diesen Rechner wieder löschen, dies geschieht mit dem `arp` Befehl. Wer keine Ethernet-Karte hat sollte diese Zeile wiederum auskommentieren.

/etc/slip.tty Soll eine dynamische Adressvergabe für SLIP-Benutzer stattfinden (angezeigt durch das Schlüsselwort `DYNAMIC` in der Adress-Spalte der Datei `/etc/slip.hosts`) so muß in der Datei `/etc/slip.tty` eine Liste aufgeführt sein, welchem seriellen Port welche IP-Adresse zugeordnet werden soll. Wer ausschließlich statische Adressen vergeben will benötigt diese Datei nicht.

In `/etc/slip.tty` sind diejenigen seriellen Devices aufgeführt, über die ein SLIP-Login möglich ist, sowie die IP-Adresse, die einem Anrufer, der sich über diese Schnittstelle einwählt, zugewiesen werden soll. Das Format dieser Datei ist wie folgt:

```
# slip.tty    tty -> IP Adressenzuweisung fuer dynamisches SLIP
# Format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Diese Tabelle legt also fest daß ein Anrufer, der sich über die Schnittstelle `/dev/ttyS0` einwählt und in der Datei `/etc/slip.hosts` einen `DYNAMIC` Eintrag hat, die Adresse `192.168.0.100` zugewiesen bekommt.

Es muß also nur eine Adresse je Schnittstelle angegeben werden, unabhängig von der Anzahl der Benutzer des dynamischen SLIP. Dadurch kann die Anzahl der verwendeten Adressen gering gehalten werden.

12.2.2 Ein SLIP-Server unter dip

Einige der Informationen aus diesem Kapitel stammen aus der `man` Online-Hilfe zu `dip`, in der ebenfalls beschrieben wird, wie mittels `dip` ein Linux SLIP-Server konfiguriert werden kann. Ausgangspunkt ist die Version `dip337o-uri.tgz`, wer eine andere Version verwendet sollte sicherheitshalber auch in der originalen Anleitung nachlesen.

`dip` besitzt auch einen speziellen Empfangsmodus, bei dem es automatisch anhand des Benutzernamens desjenigen, der `dip` gestartet hat, die serielle Leitung als SLIP-Verbindung konfiguriert, und zwar basierend auf den Informationen in der Datei `/etc/diphosts`. Dieser spezielle Modus von `dip` wird aktiviert, indem man es unter dem Namen `diplogin` startet. Zu diesem Zweck legt man einen symbolischen Link an:

```
% ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Um nun mit `dip` einen SLIP-Server zu konfigurieren müssen auch wieder spezielle Accounts angelegt werden, wobei `diplogin` als Login-Shell verwendet wird. Dies geschieht durch entsprechende Einträge in der Datei `/etc/passwd`. Als Benutzernamen für SLIP-Accounts sollte wieder die übliche Form 'S'+Rechnername verwendet werden, sodaß ein typischer Eintrag so aussehen könnte:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^          ^^         ^^  ^^   ^^   ^^   ^^
|           |          |   |    |    |    \__ diplogin als Login Shell
|           |          |   |    |    \_______ Home directory
|           |          |   |    \____________ Voller Benutzername
|           |          |   \_________________ User Group ID
|           |          \_____________________ User ID
|           \________________________________ Verschluesseltes Passwort
\____________________________________________ SLIP User Login Name
```

Wenn sich nun ein Benutzer mit dem Namen `Sfredm` einlogged und das korrekte Paßwort eingibt, wird der `login(1)`-Prozeß den Benutzer als rechtmäßig anerkennen und die die Login-Shell, also das Programm `diplogin`, starten. Dadurch daß `dip` über dem Umweg eines symbolischen Links unter dem Namen `diplogin` gestartet wird weiß das Programm automatisch, daß es als Login-Shell aufgerufen wurde. Als erstes ruft es dann die Unix-Funktion `getuid()` auf um die User-ID (UID) des Aufrufenden herauszufinden. Danach durchsucht es die Datei `/etc/diphosts` nach dem ersten Eintrag, der entweder dieser UID oder aber dem Namen des `tty` entspricht, über das die Verbindung läuft. Entsprechend den darin gegebenen Parametern wird dann die Verbindung konfiguriert.

Einfach anhand der Entscheidung, ob für einen Benutzer ein Eintrag in `diphosts` angelegt wird, oder ob die Standard-Behandlung anhand der ttys zum tragen kommt, kann ein gemischter Server aus statischen und dynamischen Adressen zusammengestellt werden.

`dip` führt weiterhin automatisch einen *Proxy-ARP* Eintrag durch, sodaß man sich darum nicht selbst kümmern muß.

/etc/diphosts In der Datei `/etc/diphosts` sind die Konfigurationsparameter für die Klienten-Rechner zusammengestellt, `dip` verwendet sie, um die SLIP-Devices richtig zu konfigurieren. Das generelle Format dieser Datei sieht so aus:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Die Felder bedeuten im Einzelnen:

1. `Login Name`: Entsprechend dem Ergebnis von getpwuid(getuid()) oder tty Name.
2. `unbenutzt`: Zur Kompatibilität mit passwd
3. `Remote Adresse`: IP Adress des einwählenden Rechners, entweder numerisch oder als Name.
4. `Lokale Adresse`: IP Adress des eigenen Rechners, ebenfalls numerisch oder als Name.
5. `Netz-Maske`: in punktierter Dezimalschreibweise.
6. `Kommentar`: Beliebig.

7. `Protokoll`: SLIP, cSLIP usw.
8. `MTU`: Dezimaler Wert.

Für alle diejenigen Benutzer, für die eine statische Adresse verwendet werden soll, muß ein solcher Eintrag in `/etc/diphosts` angefügt werden. Erlaubte SLIP-Benutzer, die ihre IP-Adresse dynamisch zugewiesen bekommen sollen werden hier **nicht** eingetragen, dadurch wird dann automatisch der Eintrag des verwendeten `tty` benutzt. Aus diesem Grund sollte man für jedes `tty`, über das sich Benutzer in das System einwählen können, ein solcher Eintrag angefügt werden um sicherzustellen, daß immer eine adäquate Konfiguration durchgeführt werden kann.

Ein Benutzer der sich in ein so konfiguriertes System einwählt sollte zunächst einen ganz normalen Login-Promt sehen. Er muß dann seinen SLIP-Benutzernamen sowie sein Paßwort eingeben. Wenn alles glatt geht bekommt er keine weitere Meldung und muß nun auf seiner Seite nur noch die Verbindung in den SLIP-Modus schalten, und die Verbindung steht.

12.2.3 SLIP Server mit dem dSLIP Paket

Matt Dillon (`dillon@apollo.west.oic.com`) hat ein Paket von kleinen Programmen und Shell-Scripts geschrieben, mit denen SLIP sowohl im Dial-in wie im Dial-out betrieben werden kann. Allerdings muß die Shell `tcsh` installiert sein, da mindestens eines der Scripts auf deren Syntax angewiesen ist. Jedoch ist dies keine große Einschränkung, da die `tcsh` bei den meisten Distributionen mitgeliefert wird. Außerdem gehört zu Matts Paket auch eine ausführbare Kopie des Programmes `expect`, das ebenfalls an einigen Stellen benötigt wird. Es ist von Vorteil wenn man sich mit `expect` bereits auskennt, da andernfalls bei der Konfiguration leicht Fehler gemacht werden können. Aus diesem Grunde empfiehlt sich das Paket mehr für die bereits mit Unix vertrauten, man sollte sich aber trotzdem nicht davon abhalten lassen, sich das Programm einmal anzusehen, zumal Matt eine sehr gute Installationsanleitung im README gibt.

Das *dSLIP* Paket bekommt man von:

```
apollo.west.oic.com:/pub/linux/dillon_src/dSLIP203.tgz
```

oder

```
sunsite.unc.edu:/pub/Linux/system/Network/serial/dSLIP203.tgz
```

Wichtig ist, das README aufmerksam zu lesen und vor allem die dort angegebenen Einträge in den Dateien `/etc/passwd` und `/etc/group` einzufügen, **bevor** ein `make install` ausgeführt wird.

12.3 Der Automounter Dämon - AMD

Dieser Abschnitt basiert auf einem Text, den Mitch DSouza zusammengestellt hat.

12.3.1 Was ist ein Automounter, warum sollte ich einen verwenden ?

Ein *Automounter* stellt ein übliches Mittel dar um ein Dateisystem (meist über NFS) nur bei Bedarf zu mounten (der Fachausdruck hierfür ist *on demand*). Dadurch wird die Belastung sowohl des Servers wie auch des Klienten verringert, außerdem bietet dieses Vorgehen ein größtes Maß an Flexibilität, auch dann, wenn es sich nicht um NFS-Dateisysteme handelt. Er bietet sogar ein Sicherheitsmechanismus an durch den ein Dateisystem von einem anderen Server angefordert und gemounted werden kann, wenn der ursprüngliche Server nicht erreicht werden kann. Durch eine besonders nützliche Art des mounts, einen sogenannten *union mount*, können sogar die Inhalte unterschiedlicher Verzeichnisse in einem einzelnen Verzeichnis zusammengefaßt werden. Um diese hochentwickelten Optionen auszunutzen ist es aber auf jeden Fall nötig, die Dokumentationen **sehr** ausführlich und genau zu lesen.

Auf einige Dinge muß besonders geachtet werden:

- Die Maps von `amd` sind **nicht** kompatibel zu denen von Sun, diese wiederum inkompatibel zu HP usw. Der Vorteil ist aber, daß `amd` ein frei erhältliches Programm ist, welches auf den angegebenen und vielen weiteren Systemen installiert werden kann. Dann fallen die Kompatibilitätsprobleme weg, und maps können auch in heterogenen Systemen gemeinsam benutzt werden. Es gibt inzwischen schon viele Beispiele von solchen gemischten Linux/Dec/NeXt/Sun usw. Netzwerken.
- Es gibt im Verzeichnis `contrib` ein `perl` Script mit dem Namen `automount2amd.pl`, mit dem die vom Sun automounter verwendeten Maps in das `amd`-Format konvertiert werden können.
- Der `portmapper` **muß** unbedingt **vor** `amd` gestartet werden.
- Für UFS Mounts gibt es **kein** Timeout.
- **Ausschließlich für Linux** wurden UFS Mounts derart erweitert, daß sie mit allen Arten von Dateisystemen (also minix, ext, ext2, xiafs...) umgehen können, wobei minix die Standardeinstellung ist. Diese undokumentierte Besonderheit kann über die `opts`-Option angesprochen werden:

  ```
  ..., opts:=type=msdos,conv=auto
  ```
- Nach Möglichkeit nie einen existierenden Verzeichnisbaum 'übermounten', und wenn, dann nur mit der `direct` Option.
- Bei Problemen **immer** die vollständige Mitprotokollierung von `amd` mit der Option `-x all` aktivieren. Ebenfalls hilfreich ist oft auch die Information die der Befehl

  ```
  % amq -ms
  ```

  liefert. Dadurch werden alle auftretenden Probleme sofort gemeldet.
- Das `getopt()` von GNU ist manchmal etwas zu schlau. Sicherheitshalber sollte deshalb '–' zwischen den `amd`-Optionen und den Parametern eingefügt werden, also z.B. so:

  ```
  % /etc/amd -x all -l syslog -a /amd -- /net /etc/amd.net
  ```

12.3.2 Woher bekomme ich AMD ?

```
sunsite.unc.edu:/pub/Linux/system/Misc/mount/amd920824upl67.tar.gz
```

Das Paket enthält sowohl Quelltexte und die vollständige Dokumentation im texinfo-Format als auch bereits kompilierte Binärdateien, die sofort installiert werden können.

12.3.3 Beispiel für eine AMD Konfiguration

Der *Automounter* wird nicht über die Datei `/etc/fstab` konfiguriert, die in einem gewöhnlichen System die Informationen über die Dateisysteme enthält. Vielmehr wird er über die Kommandozeile gesteuert.

Nehmen wir folgendes Beispiel: In der Datei `/etc/fstab` sind zwei `nfs` Dateisysteme eingetragen,

```
server-1:/export/disk  /nfs/server-1  nfs  defaults
server-2:/export/disk  /nfs/server-2  nfs  defaults
```

es werden also immer die Verzeichnisse `/nfs/server-1` und `/nfs/server-2` von den beiden Servern `server-1` und `server-2` via `nfs` gemountet. Genausogut könnte man diese beiden Einträge auskommentieren und `amd` mit dem folgenden Befehl starten:

```
/etc/amd -x all -l syslog -a /amd -- /nfs /etc/amd.server
|      | |    | |       | |     |  | |  | |             |
|      | |    | |       | |     |  | |  | |             |
'------' '----' '-------' '-----' -' '--' '-------------'
|        |      |         |       |  |    |
(1)      (2)    (3)       (4)     (5) (6) (7)
```

Die Einzelnen Komponenten bedeuten dabei

1. Den vollen Pfadnamen des ausführbaren Programmes `amd`. Befindet sich `amd` in einem Verzeichnis, das in der Variable `$PATH` enthalten ist genügt auch einfach nur `amd`.
2. `-x all`' schaltet die volle Protokollierung ein, um Fehler leichter verfolgen zu können.
3. `-l syslog` bewirkt, das die Protokollierung über den `syslog` Dämon erfolgt. Dadurch kann man getrennt entscheiden, was mit diesen Informationen geschieht, ob sie in eine freie Konsole geschrieben werden oder dauerhaft in einer Datei gespeichert werden. Anstelle von `syslog` kann auch ein Dateiname angegeben werden, dort werden dann alle Meldungen abgelegt.
4. `-a /amd` weist `amd` an, das Verzeichnis `/amd` als temporären Platz für die automounts zu verwenden. Es wird von `amd` automatisch angelegt und sollte vor dem Start von `amd` in den `/etc/rc.d/rc*` Scripts gelöscht werden.
5. `--` dient wieder dazu, der `getopt()` Funktion das Ende der Kommandozeilenoptionen mitzuteilen. Dies ist insbesondere nötig wenn man die Option `type:=` verwendet, da `getopt()` diese falsch interpretieren würde.
6. `/nfs` ist der **wahre** Mount Punkt. Auch dieser wird automatisch angelegt und sollte normalerweise **keine Unterverzeichnisse** aufweisen, wenn nicht die Option `type:=direct` verwendet wird.
7. Die *Map* für `amd` ist eine Datei, hier `/etc/amd.server`, die - für dieses Beispiel - folgende Zeilen enthält:

```
# /etc/amd.server
/defaults    opts:=rw;type:=nfs
server-1     rhost:=server-1;rfs:=/export/disk
server-2     rhost:=server-2;rfs:=/export/disk
```

Ist der `amd` Prozeß gestartet und läuft zufriedenstellend, kann der Zustand der Mount-Tabelle abgefragt werden:

```
% amq -ms
```

Ein einfaches

```
% ls /nfs
```

wird keine Dateien anzeigen. Jedoch wird ein

```
% ls /nfs/server-1
```

bewirken, daß automatisch das gesuchte Verzeichnis von dem Rechner 'server-1' über `nfs` gemountet wird, sodaß dessen Inhalt angezeigt wird! Nachdem eine einstellbare Zeit vorüber ist, innerhalb der nicht auf das Verzeichnis zugegriffen wurde, wird das Dateisystem automatisch wieder ent-mounted.

12.4 Linux als Router

Linux als Router in einem Netzwerk zu verwenden ist problemlos. Für gewöhnlich verwendet man zu diesem Zweck `gated` oder einen ähnlichen Router-Dämon, für kleinere Netzwerke können aber auch einfache statische Routes verwendet werden. Auf jeden Fall ist es aber wichtig, die folgende Frage bei der Konfiguration des Kernels zu bejahen:

```
...
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [y] y
...
```

Wer selber ein Netzwerk zu betreiben gedenkt sollte sich in jedem Fall den *Network Administrators Guide* von Olaf Kirch zulegen, dort sind auch zum Thema Routing alle notwendigen Details fachkundig zusammengefaßt.

12.5 NIS - Das Sun Network Information System

Zu diesem Thema gibt es ein eigenes HOWTO, das *NIS-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/NIS-HOWTO.html`). Wer beabsichtigt, NIS auf seinem Rechner zu verwenden wird darin über alles Notwendige informiert.

13 Experimentelle Module

Eine ganze Reihe von Programmierern entwickeln dauernd neue Erweiterungen und Module für den Netzwerk-Code von Linux. Viele sind bereits in einem sehr fortgeschrittenen Stadium, und einige sind mit der 2.0 Version auch in den Standard-Kernel aufgenommen worden. Für viele dieser Module gibt es inzwischen eigene HOWTOs:

13.1 On-Demand SLIP/PPP

Eric Schenk (`schenk@cs.toronto.edu`) hat einen Dämon geschrieben, der sowohl mit SLIP als auch mit PPP arbeitet und eine solche serielle Verbindung aufbaut, sobald ein Netzwerkdienst angefordert wird. Er setzt dabei voraus, daß ein SLIP-Device konfiguriert ist, mit dem der Dämon über ein `pty` kommuniziert. Solange die Verbindung nicht aufgebaut ist werden alle Datagramme über dieses Device geroutet. Der Dämon registriert diese und führt ein Script aus, welches die Netzwerkverbindung aufbaut und sendet dann die zwischengespeicherten Datagramme über diese Schnittstelle weiter.

Bekommen kann man diese Software von

```
sunsite.unc.edu:/pub/Linux/system/Network/serial/diald-0.14.tar.gz
```

Wichtig: Der Kernel muß unbedingt mit SLIP-Unterstützung kompiliert werden, auch wenn man PPP verwenden will. Installation und Betrieb werden von der mitgelieferten Dokumentation beschrieben.

13.2 System-V Streams

Das LiS (Linux Streams) Projekt wurde ins Leben gerufen, um unter Linux eine Implementation der System-V Streams zu entwickeln. Sie haben eine Seite im World Wide Web: die *Linux Streams Web Page* (`http://www.uc3m.es/ LiS/`). Dort kann man sich über den derzeitigen Stand der Entwicklung informieren und erfährt auch, wo es die Software gibt.

13.3 Unterstützung von ATM (Asynchronous Transfer Mode)

Auch hierzu gibt es eine Seite im World Wide Web, die von Werner Almesberger zusammengestellt wurde: *Linux-ATM* (`http://lrcwww.epfl.ch/linux-atm/`). Es gibt experimentelle Software im Alpha-Stadium, um reine ATM-Verbindungen sowie IP über ATM zu ermöglichen, man kann sie auch über die angegebene Adresse bekommen, außerdem jede Menge Information zu ATM im Allgemeinen.

Es gibt auch eine Mailing Liste zu diesem Thema, an der man teilnehmen kann, wenn man eine mail mit dem Inhalt `subscribe linux-atm` an die Adresse `majordomo@vger.rutgers.edu` sendet.

14 Diagnose-Hilfsmittel: Wenn etwas nicht funktioniert

In diesem Abschnitt werden einige der Standard-Diagnoseprogramme für das Linux Netzwerk beschrieben. Man kann sie einsetzen um die Ursache von Störungen näher einzukreisen, oder einfach nur damit spielen, um mehr über die Interna eines TCP/IP-Netzwerkes zu lernen. Einige der vorgestellten Programme sind weit mächtiger als hier beschrieben, sie in ihrem vollen Leistungsumfang zu erläutern ist aber weit außerhalb der Intention dieser Einführung. Die gegebene Information sollte aber ausreichen um diese Programme für einfache Zwecke zu benutzen.

14.1 Ping - ist da wer ?

`ping` ist Teil der NetKit-B Distribution. Das Programm sendet ein spezielles Datagramm über das Netzwerk an einen anderen Rechner. Wenn dieser eingeschaltet ist und es empfängt sendet er es wieder zurück und man weit, daß Rechner und Netzwerk in Ordnung sind. In der einfachsten Form sieht das so aus:

```
% ping gw
PING gw.vk2ktj.ampr.org (44.136.8.97): 56 data bytes
64 bytes from 44.136.8.97: icmp_seq=0 ttl=254 time=35.9 ms
64 bytes from 44.136.8.97: icmp_seq=1 ttl=254 time=22.1 ms
64 bytes from 44.136.8.97: icmp_seq=2 ttl=254 time=26.0 ms
^C

--- gw.vk2ktj.ampr.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 22.1/28.0/35.9 ms
```

`ping` sucht zunächst die richtige IP-Adresse zu dem angegebene Rechnernamen heraus (Voraussetzung ist natürlich, daß `/etc/hosts` und der Nameserver richtig konfiguriert sind) und sendet dann in regelmäßigen Abständen ein bestimmtes Datagramm an diese Adresse. Für jeden dieser `icmp echo requests` wird der andere Rechner dann ein `icmp echo reply` als Antwort zurücksenden. Jede der Zeilen `64 bytes from ...` stellt eine solche empfangene Rückantwort dar. In jeder Zeile ist auch die Adresse des Absenders dieses Paketes angegeben, die Nummer des `echo requests`, auf den mit diesem Paket geantwortet wurde, die `time to live` (Lebensdauer) sowie die `round trip time`, das ist die Zeit in Millisekunden zwischen Absendung des `request` und Empfang des entsprechenden `echo reply`. Damit ist dies in gewissen Grenzen ein Maß für die Geschwindigkeit des Netzwerkes.

`ping` würde endlos weiter seine Pakete senden, deshalb muß es abgebrochen werden. Dies geschieht durch gleichzeitiges Drücken der Tasten `Control` (Strg) und `C`. Danach gibt `ping` noch eine Statistik der gesendeten Pakete aus, dies sind die letzten beiden Zeilen: Die Anzahl der gesendeten, empfangenen sowie unbeantworteten Pakete sowie kürzeste, längste und mittlere Antwortzeit. Eine hohe Verlustrate an Paketen deutet auf Probleme mit der Verbindung hin, diese können durch schlechte Verbindungen, überlastete Router oder Kollisionen im lokalen Ethernet verursacht werden. Man kann mit `ping` die schadhafte Stelle lokalisieren, indem man nacheinander alle Knotenpunkte der Verbindung an-pingt und nachsieht, ab welchem Punkt die Verlustrate stark ansteigt.

14.2 Traceroute - Wo gehts lang ?

Mit `traceroute`, einem Teil der NetKit-A Distribution, kann man den Weg, den die Datagramme bei einer Verbindung zwischen zwei Rechnern zurücklegen, testen und anzeigen. Wie auch `ping` verwendet `traceroute` dazu das icmp Protokoll. Es verwendet jedoch einen Trick, um von jeder durchlaufenen Knotenstelle eine Rückmeldung zu bekommen, und das geht so: Das Feld `time to live` eines icmp-Datagramms dient eigentlich dazu zu verhindern, das das Datenpaket ewig im Netz herumgeschickt wird, z.B. wenn es in eine Routing-Endlosschleife gerät. Zu diesem Zweck verringert jeder Router, der das Paket weiterleitet, den Zähler in diesem Datenfeld um eins. Wird dabei der Wert Null erreicht, so wird der Router, bei dem dies geschehen ist, eine `icmp time to live expired` Meldung an den ursprünglichen Absender des Datagramms senden um ihm mitzuteilen daß das Datagramm 'abgelaufen' ist. `traceroute` sendet nun nacheinander Datagramme ab, deren `time to live`, beginnend bei eins, jeweils um eins erhöht wird. Indem es nun die eingehenden `icmp time to live expired` Meldungen auswertet kann es genau den Weg, den die Datagramme nehmen, verfolgen, bis der wirkliche Zielrechner erreicht ist. Das sieht dann so aus:

```
% traceroute minnie.vk1xwt.ampr.org
traceroute to minnie.vk1xwt (44.136.7.129), 30 hops max, 40 byte packets
 1  gw (44.136.8.97)  51.618 ms  30.431 ms  34.396 ms
 2  gw.uts (44.136.8.68) 2017.322 ms  2060.121 ms 1997.793 ms
 3  minnie.vk1xwt (44.136.7.129) 2205.335 ms  2319.728 ms  2279.643 ms
```

Die erste Spalte gibt die 'Entfernung' an, also der wievielte Knoten (Hop) in der Verbindung der bezeichnete Rechner ist. Dies entspricht dem Eintrag im `time to live` Feld des Datagramms, das vom angegebenen Rechner als expired gemeldet wurde. Der nächste Eintrag ist der Rechnername (falls dieser anhand der IP-Adresse herausgefunden werden kann) und dessen IP-Adresse. Dann kommen drei Einträge mit den Antwortzeiten für drei aufeinanderfolgende Datagramme an diesen Rechner. Der erste Punkt in der Verbindung ist also der Rechner `gw`. Seine Antwortzeiten sind sehr kurz, die Verbindung ist ein lokales Ethernet. Der nächste Router ist `gw.uts`, von dort kommen die Datagramme sehr viel langsamer zurück als von `gw`. Der Grund ist hier eine sehr langsame Verbindung über eine Funkstrecke. Der nächste Knoten ist bereits der gesuchte Rechner, `minnie.vk1xwt`. Auch hier kommen die Antworten nur sehr langsam. Dies liegt aber daran, daß die Antwortzeit jedes Rechners natürlich die Summe der Verbindungszeiten aller dazwischenliegenden Teilstrecken ist, in diesem Falle also vom eigenen Rechner zu `gw`, von dort zu `gw.uts` und schließlich von `gw.uts` zu `minnie.vk1xwt`. Berücksichtigt man diese Aufsummierung der Zeiten sieht man, daß die letzte Verbindung von `gw.uts` zu `minnie.vk1xwt` ebenfalls sehr schnell ist, diese beiden Rechner hängen also vermutlich auch an einem lokalen, schnellen Netzwerk.

Taucht bei einem `traceroute` Aufruf hinter den Zeiten ein `!N` auf so weist dies auf ein nicht erreichbares Netzwerk hin. Es ist ein Zeichen dafür, daß ein Router nicht wußte, wohin er das Datagramm weiterleiten soll. Er sendet dann ein `network unreachable` Datagramm an den Absender zurück. Meist deutet es darauf hin, daß eine Netzverbindung (zeitweise) zusammengebrochen ist. Anhand der letzten angegebenen Adresse vor dieser Fehlermeldung kann man die defekte Stelle lokalisieren.

Ebenfalls auftauchen kann ein `!H` als Hinweis darauf, daß ein Rechner (Host) nicht erreicht werden konnte. Dies bedeutet i.A. daß man bis in das lokale Netzwerk des Zielrechners gelangt ist, dieser sich aber nicht meldet weil er z.B. abgeschaltet ist.

14.3 Tcpdump - Was tut sich im Netz ?

Adam Caldwell (`acaldwel@103mort2.cs.ohiou.edu`) hat das Hilfprogramm `tcpdump` für Linux portiert. `tcpdump` kann die gesamte Aktivität auf dem Netzwerk überwachen indem es sämtlich Datagramme auf dem Weg in den eigen Rechner hinein oder heraus abhört. Damit lassen sich vom erfahrenen Administrator auch schwer zu identifizierende Netzwerkprobleme lokalisieren.

Quellen und ausführbare Programme bekommt man von

```
ftp.funet.fi:/pub/OS/Linux/PEOPLE/Linus/net-source/tools/tcpdump-3.0.3.tar.gz
```

`tcpdump` dekodiert jedes abgefangene Datagramm und zeigt es - in einer etwas kryptischen Form - als Text an. `tcpdump` bietet sich an um Protokollfehler oder plötzliche Verbindungsabbrüche aufzuspüren, denn man kann damit sehen, was auf dem Netzwerk passiert. Um es jedoch sinnvoll einsetzen zu können bedarf es eines tieferen Verständnisses der Protokolle und wie sie arbeiten, doch es kann auch einfach dazu benutzt werden um sicherzustellen, daß z.B. Datagramme den eigenen Rechner wirklich über die richtige Schnittstelle verlassen oder um zu überprüfen, ob irgendwelche Datagramme von außerhalb empfangen werden. Eine typische Ausgabe von `tcpdump` sieht etwa so aus:

```
% tcpdump -i eth0
tcpdump: listening on eth0
13:51:36.168219 arp who-has gw.vk2ktj.ampr.org tell albert.vk2ktj.ampr.org
13:51:36.193830 arp reply gw.vk2ktj.ampr.org is-at 2:60:8c:9c:ec:d4
13:51:37.373561 albert.vk2ktj.ampr.org > gw.vk2ktj.ampr.org: icmp: echo request
13:51:37.388036 gw.vk2ktj.ampr.org > albert.vk2ktj.ampr.org: icmp: echo reply
13:51:38.383578 albert.vk2ktj.ampr.org > gw.vk2ktj.ampr.org: icmp: echo request
13:51:38.400592 gw.vk2ktj.ampr.org > albert.vk2ktj.ampr.org: icmp: echo reply
13:51:49.303196 albert.vk2ktj.ampr.org.1104 > gw.vk2ktj.ampr.org.telnet: S 700506986:700506986(0) win 512 <mss 1436>
13:51:49.363933 albert.vk2ktj.ampr.org.1104 > gw.vk2ktj.ampr.org.telnet: . ack 1103372289 win 14261
13:51:49.367328 gw.vk2ktj.ampr.org.telnet > albert.vk2ktj.ampr.org.1104: S 1103372288:1103372288(0) ack 700506987 win 2048 <mss 432>
13:51:49.391800 albert.vk2ktj.ampr.org.1104 > gw.vk2ktj.ampr.org.telnet: . ack 134 win 14198
13:51:49.394524 gw.vk2ktj.ampr.org.telnet > albert.vk2ktj.ampr.org.1104: P 1:134(133) ack 1 win 2048
13:51:49.524930 albert.vk2ktj.ampr.org.1104 > gw.vk2ktj.ampr.org.telnet: P 1:28(27) ack 134 win 14335

 ..
```

`tcpdump` ohne Argumente gestartet überwacht das Netzwerk-Device mit der kleinsten Nummer, das kein Loopback-Device ist. Es kann aber - wie im Beispiel - auch ein spezielles Device angegeben werden. `tcpdump` dekodiert dann jedes Datagramm das über diese Device übertragen wird und zeigt es in einer lesbaren Form an. Der erste Eintrag ist wie leicht zu ersehen der Zeitpunkt, zu dem das Datagramm gesendet oder empfangen wurde. Der weitere Inhalt der Zeile hängt dann von der Art des Datagramms ab. Die ersten beiden Zeilen im Beispiel zeigen, wie ein arp request von albert.vk2ktj nach der Adresse von gw.vk2ktj aussieht. Die nächsten vier Zeilen sind zwei `pings` von albert.vk2ktj an gw.vk2ktj, `tcpdump` gibt dabei auch die genaue Art des `icmp` Datagramms an. Das Größer-Zeichen (>) gibt die Richtung an, in der das Datagramm übermittelt wurde, es zeigt vom Sender zum Empfänger. Die restlichen Zeilen protokollieren den Aufbau einer `telnet` Verbindung von albert.vk2ktj zu gw.vk2ktj.

Die Zahl am Ende jeden Rechnernamens gibt die verwendete Socket Nummer an, zu diesem Zweck wertet `tcpdump` die Datei `/etc/services` aus.

Alle weiteren Optionen werden in der Online-Hilfe behandelt.

Vorsicht PPP-Benutzer: Die derzeitige Version von `tcpdump` unterstützt das PPP-Protokoll nicht. Es gibt zwei Patches von Al Longyear die diese Unterstützung implementieren, es wurde jedoch bislang noch kein vollständiges `tcpdump`-Paket damit zusammengestellt. Diese Patchdateien sind auf `sunsite.unc.edu` im selben Verzeichnis wie `tcpdump`.

14.4 icmpinfo - Übersicht über empfangene icmp Meldungen

Das Internet Control Message Protocol (icmp) liefert hilfreiche Informationen über den Zustand des IP-Netzwerkes. Meist werden derartige Meldungen vom System direkt empfangen und verarbeitet, ohne daß man als Benutzer oder Netzwerkadministrator etwas davon mitbekommt. `icmpinfo` ist ein Programm, daß alle derartigen Meldungen ähnlich wie `tcpdump` mitprotokolliert und anzeigt. Laurent Demailly (`dl@hplyot.obspm.fr`) hat dieses Programm basierend auf den Quellen des BSD ping geschrieben.

Version 1.10 ist erhältlich über:

```
hplyot.obspm.fr:/net/icmpinfo-1.10.tar.gz
```

Kompilierung und Installation sind sehr einfach:

```
% cd /usr/src
% gzip -dc icmpinfo-1.11.tar.gz | tar xvf -
% cd icmpinfo-1.11
% make
```

Um `icmpinfo` zu benutzen muß man als `root` eingelogged sein. Die entschlüsselten Meldungen können wahlweise auf der Konsole ausgegeben oder via `syslog` protokolliert werden.

Um zu sehen wie `icmpinfo` arbeitet ist es sehr lehrreich, das Programm zu starten und dann ein `traceroute` auf einen entfernten Rechner auszuführen. Es werden dann die icmp-Meldungen, die `traceroute` verwendet, angezeigt.

15 FAQ - Oft gestellte Fragen und ihre Antworten

15.1 Generelle Fragen

Kann ich eine einfache Terminal-Verbindung über Modem als Netzverbindung verwenden?

Das ist möglich, und zwar mit dem TERM-Paket. TERM erlaubt es, Netzwerkverbindungen über eine normale Terminal-Sitzung aufzubauen. Es verlangt allerdings diverse Modifikationen an den Netzprogrammen, die man verwenden will. Für die meisten Anwendungsprogramme gibt es aber bereits fertig kompilierte Versionen mit TERM-Unterstützung. Näheres dazu findet sich im *TERM-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Term-HOWTO.html`).

Es dauert ewig, eine telnet/ftp/rlogin-Verbindung aufzubauen.

Der Name-Resolver ist nicht richtig konfiguriert, siehe den Abschnitt über `/etc/resolv.conf`. Es muß mindestens ein Nameserver eingetragen sein.

Ich will ein eigenes Netzwerk ohne Verbindung nach außen konfigurieren.Welche Adressen kann ich verwenden?

RFC1597 reserviert einige Adressbereiche speziell für die private Nutzung. Sie sollten unbedingt verwendet werden damit keine größeren Probleme auftreten, wenn man doch einmal Verbindung mit dem Internet bekommt. Die reservierten Adressen sind:

```
10.0.0.0        -   10.255.255.255
172.16.0.0      -   172.31.255.255
192.168.0.0     -   192.168.255.255
```

Es gibt reservierte Netzwerkadressen für die Klassen A, B und C. Man ist also in der Wahl von Größe und Design des privaten Netzwerkes nicht eingeschränkt. Da das Netzwerk nie mit dem Internet oder einem anderen Netzwerk in Kontakt tritt ist es egal, ob jemand anders dieselben Adressen verwendet. Lediglich innerhalb des eigenen lokalen Netzes müssen sie eindeutig sein.

Ich habe ein lokales Netzwerk und will von allen Rechnern aus die Modem-Internetverbindung benutzen.

Hier gibt es drei Möglichkeiten:

- Man fragt beim Internet-Provider nach, ob er anstelle einer Host-Route eine Netzwerk-Route zur Verfügung stellt. Dies ist - wenn überhaupt möglich - meist sehr teuer, stellt aber die einfachste und leistungsfähigste Variante dar.
- Man konfiguriert den Rechner, der die Modemverbindung aufbaut, als ‘SOCKS’-basierten Firewall. Dadurch wird eine proxy-Funktionalität

zur Verfügung gestellt. Die 'Außenwelt' denkt, sie würde ausschließlich mit diesem einen Rechner kommunizieren, dabei arbeitet er nur stellvertretend für die anderen Rechner im Netzwerk. Das *Firewall-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Firewall-HOWTO.html`) gibt weitere Informationen zu diesem Thema. Voraussetzung ist allerdings, daß die Anwendungsprogramme der lokalen Rechner SOCKS-fähig sind.

- Man verwendet die *IP-Masquerade* Software. Dies geht recht problemlos, da keinerlei spezielle Anwendungsprogramme verwendet werden müssen. Die Software führt lediglich alle notwendigen Adress-Übersetzungen durch sodaß alle Rechner des lokalen Netzwerkes nach außen hin erscheinen, als handle es sich um den einen Rechner mit der Modem-Verbindung. Es gibt einige Einschränkungen im Leistungsumfang dieser Methode, aber sie ist sehr viel leichter als einen kompletten Firewall zu konfigurieren.

Ich kann ftp.linux.org.uk nicht erreichen, woher bekomme ich nun die Software?

Alan's Archiv wird an verschiedenen Stellen gespiegelt, z.B.:

```
ftp.Uni-Mainz.de:/pub/Linux/packages/Net2Debugged
```

```
ftp.infomagic.com:/pub/mirrors/linux/sunacm
```

Woher weiß ich, welche Version des Kernels und/oder der Netzwerksoftware ich verwende?

Netzwerkcode und Kernel verwenden inzwischen angepaßte Versionsnummern, die Kernelversion erfährt man entweder mit

```
% uname -a
```

oder

```
% cat /proc/version
```

Wie verändere ich die Meldung, die telnet-Benutzer beim Verbindungsaufbau angezeigt bekommen?

Die normale System-Meldung steht in der Datei `/etc/issue`. Einige `telnetd` Programme verwenden für Netzwerk-Logins eine andere Datei, `/etc/issue.net`.

Ich habe eine neue Login-Meldung in /etc/issue geschrieben. Nach einem Reboot ist sie wieder verschwunden.

Einige Distributionen (Slackware) erstellen `/etc/issue` bei jeden Bootvorgang neu. Dies geschieht in der Datei `/etc/rc.d/rc.S`. Wem die dort erzeugte Meldung nicht gefällt, muß die entsprechenden Zeilen auskommentieren.

15.2 Fehlermeldungen

Ich bekomme andauernd die Meldung 'eth0: transmit timed out'. Was bedeutet das?

Normalerweise deutet es darauf hin daß das Ethernet-Kabel nicht korrekt eingesteckt ist, oder daß die Setup-Parameter der Ethernet-Karte (I/O-Adresse, IRQ usw.) falsch eingestellt sind. Man überprüfe (mit `dmesg`) die Bootmeldungen und stelle sicher, daß die Karte mit der richtigen Ethernet-Adresse gefunden wird. Ist dies der Fall liegt eventuell ein Konflikt mit einer anderen Karte im Rechner vor, z.B. eine Soundblaster, die denselben IRQ oder I/O-Port verwendet.

Wenn ich das Netzwerk benutzen will kommt die Meldung 'check Ethernet cable'.

Außer dem offensichtlichen Fall eines defekten Ethernet-Kabels kann diese Meldung auch durch eine falsch konfigurierte Ethernet-Karte hervorgerufen werden. Eventuell sollte man die Einstellungen in `/usr/src/linux/drivers/net/CONFIG` überprüfen.

15.3 Fragen zum Routing

Wieso bekomme ich die Meldung 'obsolete route request' wenn ich `route` verwenden will?

Die verwendete Version von `route` ist für den Kernel zu alt. Im Abschnitt 'Das Programmpaket zur Netzwerk-Konfiguration' ist beschrieben, woher man die aktuelle Version bekommt.

Wieso bekomme ich die Meldung 'network unreachable' wenn ich eine Verbindung herstellen will?

Die Meldung bedeutet, daß der eigene oder ein anderer Rechner nicht weiß, wohin er die Datagramme routen soll, die man an den angestrebten Zielrechner abgeschickt hat. Wenn die Meldung für alle Rechner auftritt, mit denen man sich in Verbindung setzen möchte, ist vermutlich die *Default Route* nicht oder falsch gesetzt, vergleiche dazu den Abschnitt über 'routing'.

Ich kann meinen Server/Gateway anpingen, aber keinen Rechner darüberhinaus.

Das Problem liegt vermutlich im Routing, die Konfiguration sollte nochmals anhand des entsprechenden Abschnittes 'routing' überprüft werden. Ist sie in Ordnung überprüft man als nächstes, daß der Host, mit dem man kommunizieren will, eine Route zum eigenen Rechner konfiguriert hat. Bei Modem-Verbindungen ist dies eine häufige Ursache. Es muß sichergestellt sein daß auf dem Server ein Router-Dämon wie `gated` oder `routed` läuft, und daß er ein *Proxy ARP* für die einwählenden Rechner durchführt, da sonst die Gegenseite die Datagramme zwar empfängt aber nicht weiß, wohin sie die Antworten senden soll.

15.4 Linux in Verbindung mit Novell Fileservern oder NFS

Wie benutze ich meinen vorhandenen Novell Fileserver mit meinem Linux-Rechner?

Im *IPX-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/IPX-HOWTO.html`) wird Software beschrieben, mit der diese Verbindung ohne NFS verwirklicht werden kann.

Dateien werden bei Verwendung von NFS übers Netzwerk verstümmelt.

Einige Hersteller (vornehmlich Sun) haben Rechner ausgeliefert, auf denen NFS standardmäßig ohne eine UDP Prüfsumme läuft. Dies bringt zwar Vorteile in einem Ethernet-Netzwerk, grenzt aber sonst schon an Daten-Mord. UDP Prüfsummen können auf jedem Server bei Bedarf eingeschaltet werden, Linux hat sie seit pl13 standardmäßig aktiviert. Wichtig ist, daß sie auf beiden Seiten verwendet werden.

Warum sind meine NFS-Dateien Schreibgeschützt?

Dies ist unter Linux die Standard-Einstellung für NFS-Dateisysteme. Näheres dazu steht in der `man` Online-Hilfe zu `exports(5)` und `nfsd(8)`.

15.5 Fragen zu SLIP

Was mache ich, wenn ich die IP-Adresse meines SLIP-Servers nicht kenne?

`dip` benötigt die Adresse des Servers nicht wirklich, um eine funktionierende SLIP-Verbindung aufzubauen. Die `remote`-Option wurde eingeführt, damit `dip` die `ifconfig` und `route` Befehle automatisieren kann. Wer die Adresse des Servers nicht kennt und partout nicht herausbekommt, dem hilft vielleicht ein Tip von Peter D. Junger (`Junger@samsara.law.cwru.edu`): Er gibt immer die eigene Adresse an, wenn im dip-Script nach der `remote` Adresse gefragt wird. Dies führt zu keinen weiteren Komplikationen, da die Server-Adresse nie in einem SLIP-Header auftaucht.

`dip` **funktioniert nur für root. Wie kann ich normalen Benutzern seine Verwendung ermöglichen?**

`dip` muß setuid root laufen um einige der notwendigen Dinge wie z.B. die Modifikation der Routing-Tabelle, durchzuführen. Versionen älter als dip337o enthalten aber eine Sicherheitslücke und sollten deshalb nicht setuid root laufen! In jedem Fall empfiehlt Uri Blumenthal folgendes Vorgehen:

- Anlegen einer neuen Group mit dem Namen `dip` in der Datei `/etc/group`, in die alle diejenigen Benutzer aufgenommen werden, die eine Modem-Verbindung mit `dip` aufbauen dürfen.
- Als root folgende Befehle ausführen:

```
% chown root.dip /usr/bin/dip
% chmod u=rx,g=x,o= /usr/bin/dip
% chmod u+s /usr/bin/dip
```

Ich bekomme die Meldung 'DIP: tty: set_disc(1): Invalid argument', was bedeutet das?

Normalerweise deutet es darauf hin, daß der Kernel ohne Unterstützung für SLIP kompiliert wurde. Man überprüfe, ob `/proc/net/dev` die Devices `sl0`, `sl1` usw. aufzeigt. Eine andere Möglichkeit ist, daß die verwendete Version von `dip` bereits sehr alt ist, in diesem Fall sollte man sich eine neuere besorgen.

Wenn ich einen Rechner anpinge, bekomme ich die Meldung 'wrong data byte #17...'.

In diesem Fall ist meist das Modem für das XON/XOFF Flow Protokoll konfiguriert. Für SLIP **muß** die Verbindung 8-bit-clean sein, XON/XOFF darf also nicht verwendet werden. Die andere Alternative, Hardware handshaking, funktioniert sowieso besser, man sollte also diese verwenden. Wie man sie einstellt wird im Handbuch zum Modem beschrieben.

Ich kann über SLIP zwar alle Rechner anpingen, aber ein telnet schlägt fehl.

Vermutlich liegt der Grund in einer uneinheitlichen Verwendung der Header-Kompression. Beide Seiten müssen in dieser Einstellung übereinstimmen.

Wie kann ich die Telefonverbindung beenden, wenn ich mit SLIP fertig bin?

Wer `dip` benutzt, für den genügt ein einfaches `dip -k`, dann werden alle nötigen Schritte durchgeführt. Wenn `dip` dennoch weiterläuft sollte man versuchen, es manuell mit `kill` zu beenden. Wenn der `dip`-Prozeß gestorben ist, sollte das Modem in jedem Fall aufgehängt haben. Das sicherste Vorgehen beim manuellen 'abschießen' von `dip` ist folgendes: `kill <pid>`, `kill -hup <pid>` und, wenn `dip` immernoch läuft, schließlich `kill -9 <pid>`. Dies ist übrigens nicht auf `dip` beschränkt sondern gilt für alle Unix-Prozesse.

Ich sehe eine Menge 'Overrun'-Fehler auf meinem SLIP-Port.

Die älteren Network Tools haben fälschlicherweise die komprimierten Pakete als 'Overrun' gemeldet. Dies wurde aber inzwischen behoben und sollte unter neueren Kernels und Netz-Software nicht mehr auftreten. Treten die Fehler trotz neuer Versionen auf so kann der Rechner eventuell die ankommenden Datenpakete nicht schnell genug in Empfang nehmen. Wer in seiner seriellen Schnittstelle noch keine 16550AFN UARTs hat sollte sich überlegen, aufzurüsten. 16450er, oder 8250er generieren für jedes eingehende Byte einen Interrupt und sind deshalb darauf angewiesen, daß der Prozessor seine momentane Beschäftigung unterbrechen kann um dieses Byte zu übernehmen, damit es nicht verlorengeht. Der 16550AFN dagegen hat einen 16-Byte FIFO (first in first out, ein linearer Zwischenspeicher) und generiert einen Interrupt erst kurz bevor der FIFO voll ist. Dadurch werden weniger Interrupts erzeugt und das System muß weniger Zeit für die Bedienung der seriellen Schnittstelle aufwenden. Wer sogar mehrere serielle Schnittstellen betreiben will sollte also in jedem Fall die 16550AFN UARTs verwenden.

Kann ich auch zwei SLIP Verbindungen benutzen?

Ja. Hat man zum Beispiel drei Rechner, die man verbinden will, kann man ohne weiteres auf einem der Rechner zwei SLIP-Verbindungen, je eine zu jedem der beiden anderen, konfigurieren. Die zweite Verbindung unterscheidet sich dabei in nichts von der ersten, jedoch benötigt das zweite Interface eine andere IP-Adresse als das erste. Eventuell muß man etwas mit dem Routing herumexperimentieren, aber es sollte funktionieren.

Ich habe eine Multiport I/O-Karte, wie kann ich mehr als vier SLIP-Ports verwenden?

Der SLIP-Code im Kernel verwendet eine Standardeinstellung von maximal 4 SLIP Devices. Dies wird festgelegt in der Datei `/usr/src/linux/drivers/net/slip.h`. Um den Grenzwert zu erhöhen muß er in der Definition `#define SL_NRUNIT` anstelle der dort angegebenen `4` eingesetzt werden. Weiterhin muß man die Datei `/usr/src/linux/drivers/net/Space.c` editieren und Abschnitte für `sl4`, `sl5` usw.

einfügen. Die bestehenden Definitionen können dabei als Basis verwendet werden. Nach diesen Veränderungen muß dann der Kernel neu kompiliert und installiert werden.

15.6 Fragen zu PPP

Diese werden ausführlich im *PPP-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/PPP-HOWTO.html`) behandelt.

16 Kurzanleitung: SLIP- oder PPP-Server

Einen Linux-Rechner als SLIP oder PPP Server zu konfigurieren ist vermeintlich eine leichte Aufgabe. Der eigentliche Prozeß ist sehr einfach, aber es gibt eine ganze Reihe von Aspekten der Konfiguration, und ein Verständnis darüber, wie jeder dieser Abschnitte mit den anderen verknüpft ist wird es einem hinterher ermöglichen, eventuellen Schwierigkeiten viel schneller auf die Schliche zu kommen. Doch nun zu der eigentlichen Arbeit:

1. Zusammenstellen der Hardware. IRQ und Shared-Memory Konflikte müssen vermieden werden. Jeder der seriellen Ports sollte mit einem Nullmodemkabel und einem Terminalprogramm wie `minicom` oder `seyon` auf einwandfreie Funktion geprüft werden. Zeichen müssen einwandfrei gesendet und empfangen werden. Sollen mehrere serielle Ports verwendet werden empfiehlt sich die Verwendung von Multiport-Karten und in jedem Fall 16550AFN UART's. Dies erleichtert den Umgang mit den von den seriellen Schnittstellen erzeugten Interrupts.

2. Kompilierung des Kernels. Netzwerkunterstützung muß aktiviert sein, ebenfalls *IP Forwarding*, wenn der Server auch das Routing übernehmen soll. Ohne diese Fähigkeit können die einwählenden Rechner ausschließlich mit dem Server kommunizieren. SLIP und/oder PPP müssen aktiviert sein, je nachdem welches Protokoll angeboten werden soll. CSLIP muß getrennt eingeschaltet werden, wenn es verwendet werden soll.

3. Installation und Test des neuen Kernels. `/proc/net/dev` muß die gewünschten `sl*` oder `ppp*` Devices aufzeigen.

4. Für jeden der seriellen Ports, auf dem sich Benutzer in das System einwählen können, muß ein `getty` Prozeß gestartet werden. Hilfe zu dessen Konfiguration bietet das *Serial-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/Serial-HOWTO.html`). Wichtig ist daß das Modem so eingestellt ist, daß der `DCD` Pin einen 'Carrier Detect' anzeigt, da `getty` diesen benutzt, um eingehende Anrufe zu erkennen. Eine Ausnahme stellt hier lediglich `mgetty` dar, dieses verwendet die `RING` Meldung des Modems. Deshalb muß für `mgetty` das Modem so eingestellt sein, daß es diese Meldung bei einem Anruf erzeugt. Normalerweise geschieht dies durch die Hayes-Sequenz `ATQ0V1`.

5. Der `getty` sollte - unabhängig vom SLIP/PPP Protokoll - auf Korrekte Funktion getestet werden. Bei einem normalen Einwahlversuch sollte zunächst die Loginmeldung erscheinen, und ein gewöhnlicher Login sollte möglich sein. Verschluckte oder verfälschte Zeichen bei diesem Vorgang deuten auf Probleme im Datenfluß hin.

6. Nun muß festgelegt werden, wie der Server die IP-Adressen vergibt, ob `sliplogin`, `dip` oder `dSLIP` (für SLIP-Server) verwendet werden soll usw. Die Konfiguration von `dip/diplogin` wurde bereits in Kapitel 'Ein SLIP-Server unter dip' beschrieben, die für das `dSLIP` Paket in 'SLIP Server mit dem dSLIP Paket'. Das *PPP-HOWTO* (`http://sunsite.unc.edu/mdw/HOWTO/PPP-HOWTO.html`) beschreibt die Konfiguration des `pppd` Dämons für PPP-Server.

7. Nun kann man den SLIP/PPP-Server testen, zunächst selber und dann möglichst auch von anderen: Viele Augen sehen mehr als zwei.

17 Bekannte Fehler

Der Netzwerk-Code unter Linux befindet sich in steter Entwicklung. Sicherlich enthält er noch einige Fehler, doch es werden immer weniger gemeldet. Alan Cox unterhält eine Seite im World Wide Web, die *Linux Networking News* (`http://iifeak.swan.ac.uk/NetNews.html`), in der er über den gegenwärtigen Stand der NET-3 Software informiert. Ebenfalls aktuelle Infos liefert die Datei `/usr/src/linux/net/inet/README`, die zu jeder Kernel-Distribution gehört.

18 Copyright

19 Danksagungen, Vermischtes

Der größte Dank gebührt Terry Dawson, der den englischen Originaltext in mühevoller Arbeit und von vielen ungenannten Mithelfern unterstützt zusammengestellt hat. Ohne seine Vorarbeit gäbe es keine deutsche Version.

Der Linux Netzwerk-Code hat bereits eine lange Strecke mit Höhen und Tiefen hinter sich. Die Entwickler - nicht nur die des Netzwerk-Codes, sondern alle Linux-Programmierer - haben ihr bestes gegeben um etwas zusammenzustellen das funktionell, wandlungsfähig, flexibel und vor allen Dingen frei nutzbar für uns alle ist. Wir alle sind ihnen zu Dank verpflichtet. Jeder sollte sich einmal die Zeit nehmen und die Datei `/usr/src/linux/CREDITS` durchlesen.

Peter Sütterlin (`pit@uni-sw.gwdg.de`)