

# Linux NET-3 HOWTO

---

von Terry Dawson (terry@perf.no.itg.telecom.com.au) und Peter Sütterlin (pit@uni-sw.gwdg.de)  
v1.0, Dezember 1997

Das Betriebssystem Linux enthält bereits im Kernel Unterstützung für fast alle Arten von Netzwerkprotokollen. Ziel dieses Textes ist es, die Installation und Konfiguration der Netzwerk-Software unter Linux sowie der zugehörigen Hilfsprogramme zu beschreiben.

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
1.1	Feedback . . . . .	4
1.2	Danksagung . . . . .	4
1.3	Copyright. . . . .	4
<b>2</b>	<b>Wie gehe ich mit diesem Text um?</b>	<b>4</b>
<b>3</b>	<b>Allgemeine Information über Netzwerke in Linux</b>	<b>5</b>
3.1	Eine kurze Geschichte der Netzwerk-Kernelentwicklung . . . . .	5
3.2	Wo bekomme ich weitere Informationen zum Netzwerk unter Linux? . . . . .	7
3.3	Nicht Linux-spezifische Informationsquellen. . . . .	7
<b>4</b>	<b>Grundkonfiguration</b>	<b>8</b>
4.1	Was brauche ich für den Anfang? . . . . .	8
4.1.1	Aktuelle Kernel Quelldateien. . . . .	8
4.1.2	Aktuelle Hilfsprogramme. . . . .	9
4.1.3	Anwendungsprogramme. . . . .	9
4.1.4	Adressen. . . . .	10
4.2	Wo muß ich die Konfiguration durchführen? . . . . .	11
4.3	Anlegen der Netzwerk Schnittstellen. . . . .	12
4.4	Konfiguration der Netzwerk Schnittstelle. . . . .	12
4.5	Konfiguration des Name Resolver. . . . .	14
4.5.1	Aus was besteht ein Name? . . . . .	14
4.5.2	Welche Informationen brauche ich? . . . . .	15
4.5.3	/etc/resolv.conf . . . . .	15
4.5.4	/etc/host.conf . . . . .	15
4.5.5	/etc/hosts . . . . .	16
4.6	Die Konfiguration des Loopback Interface. . . . .	16
4.7	Routing. . . . .	16
4.7.1	Was macht das routed Programm? . . . . .	18

4.8	Die Konfiguration von Netzwerk Servern und Diensten. . . . .	20
4.8.1	/etc/services . . . . .	20
4.8.2	/etc/inetd.conf . . . . .	25
4.9	Weitere Konfigurationsdateien im Netzwerkkumfeld. . . . .	27
4.9.1	/etc/protocols . . . . .	27
4.9.2	/etc/networks . . . . .	28
4.10	Netzwerksicherheit und Zugangskontrolle . . . . .	28
4.10.1	/etc/ftpusers . . . . .	29
4.10.2	/etc/securetty . . . . .	29
4.10.3	Die tcpd Hostzugangskontrolle . . . . .	29
4.10.4	/etc/hosts.equiv . . . . .	30
4.10.5	Konfiguration des ftp Dämons . . . . .	31
4.10.6	Einrichtung eines Firewall . . . . .	31
4.10.7	Weitere Tips und Vorschläge . . . . .	31
<b>5</b>	<b>Spezifische Informationen zur Netzwerk Technologie.</b>	<b>31</b>
5.1	ARCNet . . . . .	31
5.2	Appletalk (AF_APPLETALK) . . . . .	32
5.2.1	Die Konfiguration der Appletalk Software. . . . .	32
5.2.2	Exportieren eines Linux Dateisystems via Appletalk, . . . . .	33
5.2.3	Gemeinsame Nutzung eines Druckers mit Appletalk. . . . .	33
5.2.4	Starten der Appletalk Software. . . . .	33
5.2.5	Testen der Appletalk Software. . . . .	34
5.2.6	Nachteile der Appletalk Software. . . . .	34
5.2.7	Weitere Informationsquellen. . . . .	34
5.3	ATM . . . . .	34
5.4	AX25 (AF_AX25) . . . . .	34
5.5	DECNet . . . . .	35
5.6	EQL - Lastverteilung auf mehrere Leitungen. . . . .	35
5.7	Ethernet . . . . .	36
5.8	FDDI . . . . .	36
5.9	Frame Relay . . . . .	36
5.10	IP Accounting . . . . .	40
5.11	IP Aliasing . . . . .	42
5.12	IP Firewall . . . . .	42
5.13	IPX (AF_IPX) . . . . .	45
5.14	IPv6 . . . . .	45
5.15	ISDN . . . . .	45

---

5.16	IP Masquerade	46
5.17	IP Transparent Proxy	47
5.18	Mobile IP	48
5.19	Multicast	48
5.20	NetRom (AF_NETROM)	48
5.21	PLIP	49
5.21.1	Ein Kabel für PLIP.	50
5.22	PPP	51
5.22.1	Permanente Netzverbindungen mit pppd.	51
5.23	Rose protocol (AF_ROSE)	51
5.24	SAMBA - 'NetBEUI', 'NetBios' Unterstützung.	51
5.25	SLIP Klient	52
5.25.1	dip	52
5.25.2	slattach	52
5.25.3	Wann benutze ich welches Programm?	53
5.25.4	Statische SLIP Server und dip.	53
5.25.5	Dynamische SLIP Server und dip.	53
5.25.6	Die Benutzung von dip.	54
5.25.7	Dauerhafte SLIP Verbindungen mit slattach.	56
5.26	SLIP server.	57
5.26.1	SLIP Server mit sliplogin.	57
5.26.2	SLIP Server mit dip.	60
5.26.3	SLIP Server mit dem dSLIP Paket	63
5.27	Unterstützung für STRIP (Starmode Radio IP)	63
5.28	Token Ring	64
5.29	X.25	64
5.30	WaveLan Karten	64
<b>6</b>	<b>Kabel und Verkabelung</b>	<b>65</b>
6.1	Ein Serielles NULL Modem Kabel	65
6.2	Kabel für die Parallelschnittstelle (PLIP)	65
6.3	(Thin) Ethernet Verkabelung (10base2)	66
<b>7</b>	<b>Glossar der im Text verwendeten Ausdrücke</b>	<b>66</b>
<b>8</b>	<b>Linux für einen ISP ?</b>	<b>67</b>

# 1 Einleitung

Die ursprüngliche Version des NET-FAQ wurde von Matt Welsh und Terry Dawson geschrieben, bevor das Linux Documentation Project gegründet wurde. Sein Ziel war es, häufig gestellte Fragen über Linux und Netzwerke zu beantworten. Es behandelte die frühen Entwickler-Versionen der netzwerkfähigen Linux Kernel. Das NET-2-HOWTO setzte diese Aufgabe fort. Es war einer der ersten Texte des LDP und beschrieb das was zunächst als Version 2 und später als Version 3 der Linux Kernel Netzwerk Software bezeichnet wurde. Der vorliegende Text ersetzt das NET-2-HOWTO, er beschreibt ausschließlich die (aktuelle) Version 3 der Netzwerk Software im Linux Kernel.

Die früheren Versionen dieses Textes wurden ständig umfangreicher da der kleine Begriff "Netzwerk unter Linux" einen enormen Bereich abdeckt. Um diesen Umfang etwas zu reduzieren wurden etliche HOWTOs geschrieben, die sich mit spezifischen Netzwerkproblemen befassen. An den jeweiligen Stellen wird auf diese Dokumente hingewiesen, das NET-3-HOWTO selber behandelt lediglich noch solche Themen, die von den anderen HOWTOs nicht abgedeckt werden.

## 1.1 Feedback

Kommentare und vor allem aktive Beiträge zu diesem Dokument sind jederzeit willkommen. Bitte senden sie Hinweise/Kommentare zu dieser deutschen Version an mich ([pit@uni-sw.gwdg.de](mailto:pit@uni-sw.gwdg.de)).

## 1.2 Danksagung

Folgenden Personen (ohne bestimmte Reihenfolge) sei an dieser Stelle für ihre Beiträge zu diesem Text gedankt: Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor.

## 1.3 Copyright.

Dieses Dokument ist urheberrechtlich geschützt. Das Copyright für die englische *NET-3 HOWTO*, auf der dieses Dokument basiert, liegt bei Terry Dawson. Das Copyright für die deutsche Version liegt bei Peter Sütterlin.

Das Dokument darf gemäß der GNU *General Public License* verbreitet werden. Insbesondere bedeutet dieses, daß der Text sowohl über elektronische wie auch physikalische Medien ohne die Zahlung von Lizenzgebühren verbreitet werden darf, solange dieser Copyright Hinweis nicht entfernt wird. Eine kommerzielle Verbreitung ist erlaubt und ausdrücklich erwünscht. Bei einer Publikation in Papierform ist das Deutsche Linux HOWTO Projekt hierüber zu

# 2 Wie gehe ich mit diesem Text um?

Gegenüber früheren Versionen hat sich das Format dieses Dokumentes geändert. Die einzelnen Abschnitte wurden so umsortiert, daß zu Beginn alles informative Material zusammengestellt ist. Wen das nicht so sehr interessiert, der kann diesen Teil leicht überspringen. Als nächstes folgen einige generelle Bemerkungen, die man unbedingt lesen und erstehen sollte, bevor man sich mit den spezifischen Technologieteilen befaßt, die den Rest des Textes ausmachen.

### Generelle Bemerkungen

Lesen sie diesen Abschnitt unbedingt, er betrifft praktisch alle im folgenden beschriebenen Teile und ist zu deren Verständnis unbedingt notwendig.

### Welche Art Netzwerk habe ich?

Sie sollten sich darüber informieren, welche Art von Netzwerk sie installiert haben oder installieren wollen, und welche diesbezügliche Hardware in ihrem Computer eingebaut ist/wird.

### Spezifische Abschnitte

Lesen sie alle Abschnitte, die sich speziell mit der bei ihnen vorhandenen Hardware und Netztechnologie befassen. Wer genau weiß, was er will, findet hier alle für eine einzelne Technologie relevanten Daten zusammengestellt.

### Die eigentliche Konfiguration

Versuchen sie, ihr Netzwerk zu konfigurieren und notieren sie genau alle dabei eventuell auftretenden Probleme.

### Weitergehende Fragen

Stoßen sie bei der Konfiguration auf Probleme, die in diesem Text nicht behandelt werden, so lesen sie den Abschnitt über weitergehende Hilfe und Fehlermeldungen.

### Viel Spaß!

Ein funktionierendes Netzwerk macht wirklich Spaß, genießen sie es.

## 3 Allgemeine Information über Netzwerke in Linux

### 3.1 Eine kurze Geschichte der Netzwerk-Kernelentwicklung

Eine völlig neue Implementation des TCP/IP Protokolles im Kernel zu entwickeln, die mindestens so schnell ist wie bereits vorhandene war keine leichte Aufgabe. Die Entscheidung, keinen der bereits vorhandenen Treiber zu übertragen fiel zu einem Zeitpunkt an dem es einige Unsicherheiten darüber gab, ob ebendiese Implementationen mit einem restriktiven Copyright belegt werden würden. Außerdem war zu diesem Zeitpunkt der Enthusiasmus recht groß, diese Aufgabe auf eine andere Weise und womöglich sogar besser als in den vorhandenen Treibern zu lösen.

Der erste, der die Entwicklung des Linux Netzwerk-Codes leitete, war Ross Biro ([biro@yggdrasil.com](mailto:biro@yggdrasil.com)). Er schrieb einen zwar nicht ganz vollständigen aber dennoch gut brauchbaren Code, die durch einen Treiber für die WD-8003 Netzwerk-Karte vervollständigt wurde. Dies reichte aus, um viele andere dazu zu bringen, diesen Code zu testen und mit ihm zu experimentieren. Manchen ist es sogar bereits mit dieser Konfiguration gelungen, ihren Rechner an das Internet anzuschließen. Dadurch stieg im Linux-Umfeld der Druck, die Entwicklung des Netzwerk-Codes voranzutreiben. Dieser unfaire Druck, wohl zusammen mit seinen privaten Verpflichtungen, veranlaßten Ross, diese Rolle als Hauptentwickler aufzugeben. Seine Bemühungen, das ursprünglich Projekt ins Rollen zu bringen und die Verantwortung dafür zu übernehmen, daß dabei auch unter schwierigen Bedingungen etwas brauchbares herauskam wirkten als Katalysator für alle folgenden Arbeiten und sind aus diesem Grund ein wesentlicher Baustein des Erfolges des heutigen Produktes.

Die Programmierung des originalen BSD Socket Interface im Linux Kernel wurde von Orest Zborowski ([obz@kodak.com](mailto:obz@kodak.com)) durchgeführt. Dies war ein gewaltiger Schritt nach vorne da es dadurch möglich wurde, eine große Zahl von Netzwerkanwendungen ohne großen Aufwand für Linux zu portieren.

Ungefähr zu diesem Zeitpunkt schrieb Laurence Culhane ([loz@holmes.demon.co.uk](mailto:loz@holmes.demon.co.uk)) den ersten SLIP-Treiber für Linux. Dadurch konnten endlich auch solche Leute mit dem Netzwerk experimentieren, die nicht über Zugang zu einem Ethernet verfügten. Auch dieser Treiber wurde weiterentwickelt, um eine Internetanbindung über SLIP zu ermöglichen. Erneut stieg die Zahl derjenigen, die aktiv an der Erprobung und Weiterentwicklung der Netzwerk-Software mitarbeiten konnten, auch wurde vielen jetzt vor Augen geführt, was mit Linux alles möglich ist, wenn man erst einmal eine vollständige Netzwerkunterstützung hat.

Einer dieser Personen die aktiv an der Netzwerkunterstützung für Linux arbeiteten war Fred van Kempen ([waltje@uvalt.nl.mugnet.org](mailto:waltje@uvalt.nl.mugnet.org)). Nach einer Phase der Ungewißheit, die auf den Rückzug von Ross folgte, bot Fred seine Zeit und Arbeitskraft für diesen Posten an. Fred hatte einige sehr ambitionierte Pläne bezüglich der Richtung, in die die Entwicklung des Linux Netzwerk-Codes gehen sollte, und er begann damit, die notwendigen Schritte zu tun. Fred schrieb eine Version dieser Software, die als 'NET-2' Kernel Code bezeichnet wurde ('NET' Code war die Version von Ross). Diese Version konnte von vielen Leuten erfolgreich eingesetzt werden. Fred schrieb auch einige Neuerungen in die Planbücher der Entwickler, so z.B. die dynamische Geräteschnittstelle, Unterstützung

für das Amateurfunk Protokoll AX.25 sowie eine stärker modularisierte Version der Software. Freds Programme wurden zunächst von einigen Enthusiasten benutzt, deren Zahl aber ständig zunahm, je mehr sich herumsprach daß die Software gut funktionierte. Zu diesem Zeitpunkt bestand die Netzwerksoftware immer noch aus einer großen Anzahl von Patches gegenüber dem Standard-Kernel, sie gehörte nicht zur normalen Distribution. Das NET\_FAQ und die folgenden NET-2-HOWTOs beschrieben die recht komplizierte Prozedur, all dies zum Laufen zu bekommen. Freds Hauptaugenmerk lag darauf, Neuerungen zu entwickeln, und das beanspruchte Zeit. Die Nutzergemeinschaft hingegen wartete immer ungeduldiger auf eine stabile Version, die für 80% auch funktionierte. Wie bereits bei Ross stieg der Druck auf Fred als Hauptentwickler.

Alan Cox ([ialan@www.linux.uk.org](mailto:ialan@www.linux.uk.org)) schlug daraufhin eine Lösung des Problems vor. Er wollte Freds NET-2 Code nehmen und die Fehler darin beseitigen, um so eine stabile und zuverlässige Version zusammenzustellen, die die ungeduldigen Nutzer zufriedenstellte und den Druck von Freds Schultern nahm, sodaß dieser seine eigentliche Arbeit verfolgen konnte. Alan tat dies mit Erfolg, und seine erste Version wurde als 'NET-2D(ebugged)' bezeichnet. Sie arbeitete zuverlässig in vielen typischen Konfigurationen, und die Benutzer waren glücklich. Alan hatte natürlich auch eigene Ideen und auch die Fähigkeiten, die er zum Projekt beitragen wollte, und in der Folgezeit gab es viele Diskussionen darüber, in welche Richtung die Entwicklung des Netzwerk-Codes gehen sollte. Es bildeten sich zwei Lager in der Linux-Gemeinde. Die eine vertrat die Ansicht, der Code müsse zunächst funktionieren, dann könne man ihn verbessern, die andere Gruppe wollte ihn zunächst verbessern. Linus fällt letztendlich die Entscheidung, indem er Alan seinen Unterstützung anbot und seine Version in die offiziellen Kernel Distribution aufnahm. Dadurch geriet Fred in eine schwierige Lage. Jede Weiterentwicklung seines Codes hätte nun nicht mehr die Verbreitung und breite Nutzerbasis, die für ein gutes Testen nötig wäre. Dadurch würde der Fortschritt langsam und schwierig werden. Fred arbeitete noch eine zeitlang weiter, zog sich dann aber zurück und Alan wurde der neue Kopf der Netzwerk Entwicklung.

Donald Becker ([becker@cesdis.gsfc.nasa.gov](mailto:becker@cesdis.gsfc.nasa.gov)) zeigte bald sein Talent auf dem Bereich des Low-Level Netzwerk-Codes und schuf eine große Zahl von Ethernet-treibern; fast alle in der Standard Kernel Distribution enthaltenen wurden von ihm entwickelt. Auch einige andere Personen haben wichtige Beiträge geliefert, doch Donalds Arbeit war äußerst fruchtbar und rechtfertigt so die besondere Erwähnung.

Alan setzte seine Arbeit an der Verbesserung des NET-2D Codes fort, und beschäftigte sich mit einigen Bereichen der 'TODO' Liste, die bislang unberücksichtigt geblieben waren. Mit der Stabilisierung der Kernelversionen der 1.3 Serie hatte auch der Netzwerk-Code den Schritt zur Version NET-3 vollzogen, auf dem auch die aktuellen Versionen basieren. Alan arbeitete an unterschiedlichen Aspekten des Netzwerk-Codes und mit der Hilfe einer Zahl anderer talentierter Programmierer aus der Linux Gemeinschaft wuchs der Code in alle möglichen Richtungen. Alan schrieb die dynamischen Netzwerk Devices und die ersten standardkonformen AX.25 und IPX Implementationen. Seine Feinarbeit am Code hat Alan fortgesetzt und in auf den heutigen Stand verbessert.

Unterstützung für PPP wurde von Michael Callahan ([callahan@maths.ox.ac.uk](mailto:callahan@maths.ox.ac.uk)) und Al Longyear ([longyear@netcom.com](mailto:longyear@netcom.com)) implementiert. Auch dies war ein wichtiger Schritt der die Anzahl derjenigen Nutzer erhöhte, die Linux für Netzwerkaufgaben einsetzen.

Durch Jonathon Naylor ([jsn@cs.nott.ac.uk](mailto:jsn@cs.nott.ac.uk)) wurde der AX.25 Code von Alan deutlich verbessert und Unterstützung für das NetRom Protokoll hinzugefügt. Damit war Linux das einzige System das sich rühmen konnte, von Haus aus AX.25/NetRom zu unterstützen.

Selbstverständlich haben darüberhinaus hunderte weiterer Personen wichtige Beiträge zur Weiterentwicklung der Netzwerk-Software für Linux geliefert. Einige dieser Namen werden weiter unten in den entsprechenden Abschnitten erwähnt, andere haben Module oder Treiber geschrieben, Fehler beseitigt, Vorschläge gemacht, Tests durchgeführt oder einfach moralische Unterstützung geliefert. Jeder von ihnen kann von sich sagen, seinen Teil zum Ganzen hinzugefügt zu haben. Der Linux Netzwerk-Code ist ein hervorragendes Beispiel dafür, welche beeindruckende Ergebnisse der Linux-typische anarchische Stil der Entwicklung liefern kann. Und diese Entwicklung geht natürlich noch immer weiter.

## 3.2 Wo bekomme ich weitere Informationen zum Netzwerk unter Linux?

Alan Cox, der derzeit die Entwicklung des Netzwerk Codes leitet, unterhält eine Seite im World Wide Web, die die Highlights der derzeitigen Entwicklung auflistet:

```
http://www.uk.linux.org/NetNews.html
```

Eine andere gute Quelle ist das Buch von Olaf Kirch: *The Network Administrators Guide*. Dieser ist ein Teil des Linux Documentatation Project und kann entweder via ftp in diversen Formaten besorgt werden

```
sunsite.unc.edu:/pub/Linux/docs/LDP/network-guide/
```

oder inzwischen auch direkt über das Netz gelesen werden:

```
http://sunsite.unc.edu/LDP/LDP/nag/nag.html
```

Olafs Buch ist sehr verständlich und gibt einen sehr tiefgehenden Einblick in die Netzwerk Konfiguration unter Linux.

Unter den Linux Newsgroups gibt es auch eine, die sich speziell mit allen Belangen des Netzwerkes befaßt: `de.comp.os.linux.networking`.

Weiterhin besteht eine Mailing Liste zum Thema Netzwerke. Um sie zu abonnieren genügt eine kurze Mail:

```
To: majordomo@vger.rutgers.edu
Subject: anything at all
Message:
```

```
subscribe linux-net
```

Auf vielen der diversen IRC Netzwerke gibt es auch oft #linux Kanäle. Dort ist meist auch jemand bereit und in der Lage, Hilfestellungen zum Thema Netzwerke zu geben.

Eines sollte man aber immer beherzigen, wenn man mit seinen Problemen an die Öffentlichkeit will: Es sollte immer soviel wie möglich an *relevanter* Information angegeben werden. Insbesondere sind das die Versionsnummern des Kernels und der Software wie z.B. *pppd* oder *dip*, sowie eine genaue Beschreibung der auftretenden Probleme. Dies umfaßt auch den genauen Wortlaut etwaiger Fehlermeldungen sowie die genaue Syntax, mit der man ein Programm startet.

## 3.3 Nicht Linux-spezifische Informationsquellen.

Wer nach einer grundlegenden Einführung in TCP/IP Netzwerke sucht, dem seien folgende Dokumente empfohlen:

### TCP/IP introduction

entweder als Textversion (`athos.rutgers.edu:/runet/tcp-ip-intro.doc`)

oder in PostScript Form (`athos.rutgers.edu:/runet/tcp-ip-intro.ps`)

### TCP/IP administration

entweder als Textversion (`athos.rutgers.edu:/runet/tcp-ip-admin.doc`)

oder in PostScript Form (`athos.rutgers.edu:/runet/tcp-ip-admin.ps`)

Noch detailliertere Informationen zum TCP/IP Netzwerk findet man in folgendem Buch:

"Internetworking with TCP/IP"  
von Douglas E. Comer

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

Das folgende Buch befaßt sich mit dem Schreiben von Netzwerk-Anwendungen in einer Unix Umgebung:

"Unix Network Programming"  
von W. Richard Stevens

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

Ein guter Tip ist eventuell auch die Newsgroup `comp.protocols.tcp-ip`.

Eine ungemein wichtige Quelle für spezielle technische Informationen zu Internet und TCP/IP Netzwerken sind die RFC's. RFC ist ein Akronym für 'Request For Comment', also 'Bitte um Kommentar'. Es handelt sich dabei um den Standard, in dem Internet Protokolle dokumentiert werden. Es gibt viele Stellen, an denen RFC's gesammelt und archiviert werden. Viele davon sind über `ftp` erreichbar, manche bieten Zugang über das WWW, dann oft gekoppelt mit Suchmaschinen, mit denen eine gezielte Stichwortsuche möglich ist.

Eine mögliche Anlaufstelle ist die Nexor RFC database:

<http://pubweb.nexor.co.uk/public/rfc/index/rfc.html>

## 4 Grundkonfiguration

Die folgenden Abschnitte sollte man gut durchlesen, denn ihr Verständnis ist sehr wichtig, bevor man mit der tatsächlichen Konfiguration beginnen kann. Es handelt sich um grundlegende Prinzipien die unabhängig davon sind, welche Art von Netzwerk letztendlich verwendet wird.

### 4.1 Was brauche ich für den Anfang?

Einige Dinge benötigt man, bevor man sich mit der Zusammenstellung und Konfiguration seines Netzwerkes beschäftigen kann. Die wichtigsten davon sind:

#### 4.1.1 Aktuelle Kernel Quelldateien.

Der derzeit installierte Kernel hat oft nicht die Treiber für die gewünschte Hardware und Netzwerk-Protokolle eingebunden. Hier ist eine Neukompilation des Kernels mit den entsprechenden Optionen notwendig.

Die jeweils aktuelle Kernel-Version befindet sich auf

`ftp.funet.fi:/pub/Linux/PEOPLE/Linus/v2.0`

Normalerweise wird das Archiv mit den Quelltexten in das Verzeichnis `/usr/src/linux` entpackt. Weiterführende Informationen dazu, wie man Patches einspielt und den Kernel übersetzt, findet man im *Kernel HOWTO*. Die Konfiguration der verschiedenen Module beschreibt das *Module HOWTO*

Solange nicht besonders auf eine besondere Kernel-Version verwiesen wird, sollte man bei den Standard-Kernels bleiben. Dies sind die Versionen mit gerader zweiter Ziffer. Die Entwickler-Kernel (gekennzeichnet durch eine ungerade zweite Ziffer, derzeit 2.1.xx) können strukturelle Veränderungen oder Test-Code enthalten, die im Zusammenspiel mit anderen Programmen des Systems zu Problemen führen können. Wer sich nicht sicher ist, daß er mit derartigen Schwierigkeiten umgehen kann, sollte bei den Standard-Versionen bleiben.



### 4.1.2 Aktuelle Hilfsprogramme.

Diese Programme (englisch: network tools) dienen dazu, die Netzwerk Devices (Kernel-Schnittstellen zur Hardware) zu konfigurieren. Damit können also zum Beispiel Netzwerkadressen zugeordnet werden oder routes definiert werden.

Normalerweise kommen alle neueren Linux Distributionen mit diesen Hilfsprogrammen. Wer sie bei der Erstinstallation weggelassen hat, muß dies in jedem Fall nachholen.

Wer keine fertige Distribution verwendet muß sich die Quellen selbst besorgen und die nötigen Programme kompilieren. Dies ist aber nicht weiter schwierig.

Sie werden betreut von Bernd Eckenfels und können via ftp bezogen werden, entweder von

```
ftp.inka.de:/pub/comp/Linux/networking/NetTools/
```

oder die Spiegelung

```
ftp.linux.uk.org:/pub/linux/Networking/PROGRAMS/NetTools/
```

Auf jeden Fall muß man sich vergewissern, daß die Version sich auch mit dem eingesetzten Kernel verträgt. Um die gegenwärtig (also als dieser Text geschrieben wurde) aktuelle Version zu installieren geht man wie folgt vor:

```
#
# cd /usr/src
# tar xvfz net-tools-1.32-alpha.tar.gz
# cd net-tools-1.32-alpha
# make config
# make
# make install
#
```

Wer außerdem auch beabsichtigt, ein Firewall aufzusetzen oder IP Masquerading zu verwenden, wird darüberhinaus das Programm *ipfwadm* benötigen. Die aktuellste Version bekommt man über `ftp.xos.nl:/pub/linux/ipfwadm`. Auch dort gibt es unterschiedliche Versionen, man muß deshalb darauf achten die für den eigenen Kernel passende auszuwählen.

Die Installation ist dann ganz einfach (wieder am Beispiel der gerade aktuellen Version):

```
#
# cd /usr/src
# tar xvfz ipfwadm-2.3.0.tar.gz
# cd ipfwadm-2.3.0
# make
# make install
#
```

### 4.1.3 Anwendungsprogramme.

Mit 'Anwendungen' sind hier Programme wie *telnet* oder *ftp* sowie die zugehörigen Server gemeint. David Holland (`dholland@hcs.harvard.edu`) verwaltet inzwischen eine Distribution mit den am weitesten verbreiteten dieser Programme. Man erhält sie hier:

```
ftp.uk.linux.org:/pub/linux/Networking/base
```

Zur Installation der derzeit aktuellen Version geht man folgendermaßen vor:

```

#
# cd /usr/src
# tar xvfz /pub/net/NetKit-B-0.08.tar.gz
# cd NetKit-B-0.08
# more README
# vi MCONFIG
# make
# make install
#

```

#### 4.1.4 Adressen.

Die reinen Internet Protokoll Adressen bestehen aus 4 Bytes. Standardmäßig schreibt man diese in einer durch Punkte getrennten Dezimalschreibweise: Jedes Byte wird in eine Dezimalzahl umgewandelt (0-255), wobei führende Nullen weggelassen werden (außer wenn die Zahl Null ist). Diese vier Zahlen werden dann, durch einen Dezimalpunkt '.' getrennt, hintereinander aufgeschrieben. Für gewöhnlich bekommt jedes Interface eines Rechners eine eigene IP Adresse. Es ist zwar erlaubt, daß verschiedene Schnittstellen eines Rechners dieselbe Adresse verwenden, dies wird aber normalerweise nicht gemacht.

Ein Internet Protokoll Netzwerk besteht aus einer fortlaufenden Sequenz von IP Adressen. Alle Adressen innerhalb eines solchen Netzwerkes haben einige Ziffern mit den anderen gemeinsam. Dieser Übereinstimmende Teil wird als 'Netzwerk-Anteil' bezeichnet, die verbleibenden Ziffern bilden den Host-Anteil (Rechner-Teil). Die Anzahl an Bits die bei allen Adressen im Netz gleich ist, bezeichnet man als *netmask*. Ihre Aufgabe ist es festzustellen, ob ein Rechner zu diesem Netzwerk gehört, oder nicht. Hier ein Beispiel:

```

-----
Host Address      192.168.110.23
Network Mask     255.255.255.0
Network Portion  192.168.110.
Host portion          .23
-----
Network Address  192.168.110.0
Broadcast Address 192.168.110.255
-----

```

Jede Adresse, die bitweise mit der netmask durch ein logisches AND verknüpft wird, ergibt so die Adresse des Netzwerkes, zu der sie gehört. Die Netzwerk-Adresse besitzt deshalb immer die kleinste Nummer aus dem Bereich des Netzwerkes, und der Host-Anteil ist immer Null.

Die Broadcast Adresse ist eine besondere Adresse, auf die jeder Rechner eines Netzwerkes zusätzlich zu seiner eigenen, eindeutigen reagiert. An diese Adresse werden Datagramme gesendet, die jeder Rechner im Netzwerk erhalten soll. Einige besondere Datentypen wie Routing-Informationen oder Warnungen werden über diese Broadcast Adresse verbreitet, damit alle Rechner sie sofort erhalten. Es gibt zwei verwendete Standards dafür, wie eine solche Broadcast-Adresse aussieht. Am weitesten verbreitet ist es, die höchste Nummer des Adressbereiches zu verwenden, im obigen Beispiel also die 192.168.110.255. An einigen Stellen wird auch die Netzwerk-Adresse für diesen Zweck verwendet. In der Praxis ist es egal, welche dieser Adressen verwandt wird. Es muß nur sichergestellt sein, daß alle Rechner des Netzes mit derselben Broadcast-Adresse konfiguriert werden.

In einer recht frühen Phase der Entwicklung des IP Protokolles wurden einige willkürlich gewählte Bereiche des Adressraumes zu Netzwerken zusammengefaßt, die man als Klassen bezeichnet. Diese Klassen stellen die Standard-Größen für ein Netzwerk dar, die Aufteilung ist wie folgt:

Netzwerk Klasse	Netmask	Netzwerk Adressen
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 - 239.255.255.255

Welcher dieser Adressen man verwenden sollte hängt vom jeweiligen Fall ab, eventuell muß man eine Kombination der unten aufgeführten Aktionen durchführen.

#### Anbinden eines einzelnen Linux-Rechners in ein bestehendes Netz

In diesem Fall muß man sich an den Administrator des Netzes wenden und ihn um folgende Informationen bitten:

- IP Adresse für den Rechner
- IP Netzwerk Adresse
- IP Broadcast Adresse
- IP Netmask
- Adresse des Routers
- Adresse des Domain Name Servers

Mit diesen Angaben kann man dann sein Netzwerk unter Linux konfigurieren.

#### Neubildung eines Netzwerkes, daß keine Verbindung zum Internet hat

Wer sich ein kleines privates Netzwerk zulegen will und nicht beabsichtigt, dieses jemals mit dem Internet zu verbinden, kann im Prinzip seine Adressen völlig frei auswählen. Aus Sicherheitsgründen, und um trotzdem eine gewisse Konsistenz in der Adressenvergabe zu wahren, wurden jedoch einige Bereiche des Adressraumes speziell für diesen Zweck reserviert. Sie sind im RFC1597 festgelegt:

Adressbereiche f. private Nutzung		
Netzwerk Klasse	Netmask	Netzwerk Adressen
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

Zuerst sollte man sich überlegen, wie groß das eigene Netz sein soll und dann einen geeigneten Bereich auswählen.

## 4.2 Wo muß ich die Konfiguration durchführen?

Unter Linux gibt es unterschiedliche Ansätze, wie die Boot-Prozedur abläuft. In jedem Fall wird aber, nachdem der Kernel geladen ist, ein Programm mit dem Namen *init* gestartet. *init* liest dann die Konfigurationsdatei */etc/inittab* und beginnt den eigentlichen Boot-Prozeß. Es gibt verschiedene Versionen des *init*-Programmes, und das ist auch bereits der Hauptgrund für die unterschiedlichen Bootkonzepte der verschiedenen Distributionen.

Für gewöhnlich enthält */etc/inittab* einen Eintrag der Form

```
si::sysinit:/etc/init.d/boot
```

Diese Zeile legt den Namen desjenigen Shell-Scriptes fest, das den Bootprozeß steuert. In gewisser Weise handelt es sich um das Äquivalent zu der Datei AUTOEXEC.BAT in DOS.

Meist werden von diesem Script aus weitere Scripte aufgerufen, und oft ist eines davon dann für die Konfiguration des Netzwerkes zuständig.

Die folgende Tabelle gibt einen Anhaltspunkt, welche Dateien das für die diversen Distributionen sind:

Distrib.	Schnittstellen Konfig./Routing	Server Initialisierung
Debian	/etc/init.d/network	/etc/init.d/netbase /etc/init.d/netstd_init /etc/init.d/netstd_nfs /etc/init.d/netstd_misc
Slackware	/etc/rc.d/rc.inet1	/etc/rc.d/rc.inet2
RedHat	/etc/sysconfig/network-scripts/ifup-<ifname>	/etc/rc.d/init.d/network

Die meisten modernen Distributionen stellen ein Programm zur Verfügung, mit dem man die gängigsten Netzwerk Schnittstellen konfigurieren kann. Es lohnt sich auf jeden Fall diese Programme auszuprobieren, bevor man sich an eine manuelle Installation macht.

Distrib	Netzwerk Konfigurations Programm
RedHat	/sbin/netcfg
Slackware	/sbin/netconfig

### 4.3 Anlegen der Netzwerk Schnittstellen.

In vielen Varianten von Unix haben die verschiedenen Netzwerk Devices feste Einträge im Verzeichnis /dev. Nicht so bei Linux. Hier werden diese Einträge dynamisch von der Software angelegt, die entsprechenden Dateien in /dev müssen also nicht vorhanden sein.

In fast allen Fällen werden die Device-Einträge für das Netzwerk automatisch von den jeweiligen Treibern angelegt, sobald diese bei der Initialisierung die entsprechende Hardware vorfinden. So legt der Ethernet-Treiber z.B. die Schnittstellen eth[0..n] an, durchnummeriert in der Reihenfolge, in der die Hardware gefunden wird. Der ersten Karte wird der Eintrag eth0 zugeordnet, der zweiten eth1 usw.

Manche Device-Einträge, insbesondere SLIP und PPP, werden jedoch erst aufgrund von Benutzerprogrammen angelegt. Auch in diesem Fall gilt die sequentielle Durchnummerierung, sie werden eben nur nicht bereits beim Booten des Systems angelegt. Das liegt daran, daß sich die Anzahl der aktiven slip oder ppp Geräte bei laufendem Betrieb ändern kann - im Gegensatz zu Ethernetkarten. Diese Fälle werden später genauer behandelt.

### 4.4 Konfiguration der Netzwerk Schnittstelle.

Hat man sich alle benötigten Programme und Informationen besorgt, kann mit der Konfiguration der Netzwerk Schnittstelle begonnen werden. Mit Konfiguration ist dabei gemeint, der Schnittstelle die Adresse zuzuteilen sowie für andere

einstellbare Parameter die richtigen Werte einzusetzen. Das hierfür am meisten verwendete Programm ist `ifconfig`, das steht für Interface Configure, also Schnittstellenkonfiguration.

Ein typisches Beispiel für den Einsatz von `ifconfig` ist etwa

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

In diesem Fall wird die Schnittstelle `eth0` mit der Adresse `192.168.0.1` sowie der Netmask `255.255.255.0` konfiguriert. Das abschließende `up` aktiviert die Schnittstelle.

Der Kernel hat einige voreingestellte Standardwerte für viele der Konfigurationsparameter. So kann man natürlich Netzwerkadresse und Broadcastadresse für eine Schnittstelle festlegen. Tut man dies nicht, wie im obigen Beispiel, dann versucht der Kernel für diese Parameter vernünftige Werte anzunehmen. Dies macht er anhand der Netzwerk-Klasse der angegebenen IP-Adresse. In diesem Fall wäre das ein Klasse-C Netz, dementsprechend benutzt der Kernel `192.168.0.0` als Netzwerk-Adresse und `192.168.0.255` als Broadcast-Adresse.

`ifconfig` besitzt unzählige Parameter. Die wichtigsten davon sind

**up**

aktiviert die Schnittstelle

**down**

deaktiviert die Schnittstelle

**[ - ]arp**

(de-)aktiviert das Adress-Auflösungs-Protokoll. (address resolution protocol) für diese Schnittstelle.

**[ - ]allmulti**

(de-)aktiviert den sogenannten promiscuous mode. In diesem Modus kann man eine Schnittstelle anweisen auch Datenpakete anzunehmen, die nicht an diese Schnittstelle adressiert sind. Dies ist sehr wichtig für Programme wie `tcpdump`.

**mtu N**

legt die MTU fest.

**netmask addr**

legt die Netzwerk Maske für die Schnittstelle fest.

**irq addr**

legt den verwendeten Interrupt der Hardware fest. Dies funktioniert aber nur für einige wenige Geräte.

**[ - ]broadcast [addr]**

damit kann die Adresse für Broadcast-Meldungen festgelegt werden oder die Annahme solcher Pakete abgeschaltet werden.

**[ - ]pointpoint [addr]**

hiermit wird die Adresse des Rechners am anderen Ende der Verbindung festgelegt, z.B. im Falle von SLIP und PPP Verbindungen.

**hw <type> <addr>**

damit lassen sich für bestimmte Netzwerk-Typen die Hardware Adressen festlegen. Das ist für Ethernet kaum nützlich, für andere Typen wie AX.25 aber schon.

`ifconfig` kann im Prinzip zur Konfiguration jeder beliebigen Netzwerk Schnittstelle verwendet werden. Einige Programme wie `pppd` oder `dip` machen dies jedoch selbständig, sodaß sich ein manueller Aufruf von `ifconfig` in diesem Fall erübrigt.

## 4.5 Konfiguration des Name Resolver.

Der *Name Resolver* ist ein Teil der Standardbibliothek von Linux. Seine Aufgabe ist es, benutzerfreundliche Rechnernamen wie `ftp.funet.fi` in Rechnerfreundliche IP-Adressen wie `128.214.248.6` zu übersetzen.

### 4.5.1 Aus was besteht ein Name?

Jeder ist wohl inzwischen mit Rechnernamen im Internet vertraut, doch mancher versteht nicht genau wie sie gebildet werden. Namen im Internet haben eine hierarchische Struktur, bilden also so etwas wie einen Baum mit Verästelungen. Eine *Domain* ist eine Gruppe von Namen. Eine solche *Domain* kann wiederum unterteilt sein in mehrere *subdomains*. Eine *toplevel domain* ist eine *domain*, die nicht mehr *subdomain* einer anderen ist. Diese *Toplevel Domains* sind im RFC-920 festgelegt. Beispiele für die bekanntesten *Toplevel Domains* sind:

#### COM

Kommerzielle Organisationen

#### EDU

Bildung und Lehre

#### GOV

Regierungsstellen

#### MIL

Militärische Organisationen

#### ORG

Andere Organisationen

#### Länderkennzeichen

Diese sind gebildet aus zwei Buchstaben, die für ein Land stehen country.

Jede dieser höchsten Domänen hat nun Unterdomänen. So gibt es für viele Länder wieder eine Unterteilung entsprechend der höchsten Domänen, also etwa `com.au` und `gov.au` für kommerzielle und staatliche Organisationen in Australien. Aus historischen Gründen liegen praktisch alle nicht länderspezifischen Toplevel Domänen in den USA, obwohl auch diese einen spezifischen Länder-Code (`.us`) besitzen.

Die nächste Ebene der Unterteilung stellt meist der Name der Organisation dar. Weitere Unterdomänen sind dann sehr unterschiedlich, oft basieren sie auf internen Strukturen der jeweiligen Organisation, jedoch kann der Netzadministrator jedes ihm sinnvoll erscheinende Kriterium zur Unterteilung verwenden.

Der erste, am weitesten links stehende Teil des Namens ist immer der eindeutige Name des jeweiligen Rechners, man bezeichnet ihn als *hostname*, der übrige Teil rechts davon wird *domainname* genannt. Beide zusammen bilden den *Fully Qualified Domain Name*.

Am Beispiel meines eigenen Email Rechners ist der vollständige Domainname `perf.no.itg.telstra.com.au`. das heißt der Rechnername (*hostname*) ist `perf`, der *domainname* `no.itg.telstra.com.au`. Die oberste Domain (*top level domain*) ist Australien (`.au`) und es handelt sich um eine kommerzielle Organisation (`.com`). Der Name der Firma ist `telstra`, und die Namensgebung der internen Unterdomänen basiert auf der Firmenstruktur, in diesem Fall befindet sich der Rechner in der Information Technology Group, Sektion Network Operation.

#### 4.5.2 Welche Informationen brauche ich?

Natürlich muß man wissen, zu welcher Domäne der Rechner gehören soll. Weiterhin benötigt die Software, die das Übersetzen von Namen in gültige IP-Adressen übernimmt, die Adresse eines *Domain Name Servers*, dessen IP-Nummer man sich ebenfalls besorgen muß.

Es gibt insgesamt drei Dateien, die editiert werden müssen, auf jede von ihnen wird im folgenden eingegangen.

#### 4.5.3 `/etc/resolv.conf`

`/etc/resolv.conf` ist die zentrale Konfigurationsdatei für den Name Resolver. Das Format ist sehr einfach, es ist eine Textdatei mit einem Schlüsselwort pro Zeile. Normalerweise werden drei davon benutzt, dies sind:

##### **domain**

Dieser Eintrag bestimmt den Namen der lokalen Domain.

##### **search**

Mit diesem Eintrag kann man die Namen von zusätzlichen Domänen angeben, in denen nach einem Hostnamen gesucht wird.

##### **nameserver**

Mit diesem Eintrag - es können mehrere davon angegeben werden - gibt man die IP Adresse eines Domain Name Servers an.

Eine typische Datei `/etc/resolv.conf` sieht etwa so aus:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

In diesem Beispiel ist der Standard Domain Name, der an nicht vollständige angegebene Rechnernamen angehängt wird, `maths.wu.edu.au` ist. Wird der Rechner in dieser Domain nicht gefunden, wird auch in der Domäne `wu.edu.au` gesucht. Weiterhin sind zwei unabhängige Nameserver Einträge vorhanden, beide können von der Name Resolver Software benutzt werden um Namen aufzulösen.

#### 4.5.4 `/etc/host.conf`

In der Datei `/etc/host.conf` können einige Verhaltensweisen der Name Resolving Software festgelegt werden. Das Format dieser Datei ist ausführlich in der Online Hilfe zu `resolv+(8)` beschrieben. Jedoch wird in praktisch allen Fällen das folgende Beispiel ausreichend sein:

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

Mit diesen Einträgen wird festgelegt, daß die Software zunächst in der Datei `/etc/hosts` nach einer Namen - Adressen Zuordnung sucht, bevor der Nameserver gefragt wird. Außerdem sollen alle gültigen Adresseinträge, die in `/etc/hosts` gefunden werden, als Antwort geliefert werden, und nicht nur der erste.

#### 4.5.5 /etc/hosts

In der Datei `/etc/hosts` können die IP Adressen von lokalen Rechnern eingetragen werden. Ein Rechner, dessen Namen in dieser Datei auftaucht, wird auch ohne eine Nachfrage bei dem Domain Name Server gefunden. Der Nachteil dabei ist aber, daß man diese Datei selber auf dem aktuellen Stand halten muß, wenn sich die IP Adresse eines hier eingetragenen Rechners ändert. In einem gut verwalteten System wird man hier meist nur Einträge für das Loopback Interface sowie den lokalen Rechnernamen vorfinden:

```
# /etc/hosts
127.0.0.1      localhost loopback
192.168.0.1   this.host.name
```

Wie man am ersten Eintrag sieht, sind auch mehrere Namen je Adresseintrag erlaubt.

### 4.6 Die Konfiguration des Loopback Interface.

Das loopback Interface ist eine spezielle Schnittstelle, über die man eine Verbindung zum eigenen Rechner aufbauen kann. Es gibt einige Gründe, warum dies sinnvoll sein kann, zum Beispiel wenn man Netzwerk Software testen will, ohne dabei von anderen Teilnehmern des Netzes gestört zu werden. Die Standard IP Adresse für dieses Loopback Interface ist `127.0.0.1`. Unabhängig, auf welchem Rechner man arbeitet, ein `telnet 127.0.0.1` baut immer eine Verbindung zum lokalen Rechner auf.

Die Konfiguration dieser Schnittstelle ist äußerst einfach und sollte auf jeden Fall vorgenommen werden:

```
# ifconfig lo 127.0.0.1
# route add -host 127.0.0.1 lo
```

Der `route`-Befehl wird im nächsten Kapitel ausführlich behandelt.

### 4.7 Routing.

*Routing* ist ein wichtiges Thema, es ließen sich leicht Bände damit füllen. Obwohl die meisten nur recht geringe Ansprüche an das Routing haben, trifft das für einige nicht zu. Im folgenden werden nur die grundlegenden Aspekte des Routing behandelt. Wer weitergehende Informationen zu diesem Thema benötigt, der sei auf die Literaturhinweise zu Beginn dieses Dokumentes verwiesen.

Zunächst zum Begriff selber: Was ist *IP Routing*? Hier ist die Definition, die ich selber verwende:

IP Routing ist der Prozeß über den ein Rechner mit unterschiedlichen Netzwerkanbindungen entscheidet, über welche Verbindung ein empfangenes IP Datagramm weitergeleitet werden soll.

Dies soll an einem Beispiel eines typischen Routers in einem Büro verdeutlicht werden. Dieser habe eine PPP Verbindung zum Internet, bedient über einige Ethernet Segmente lokale Workstations und ist über eine weitere PPP Verbindung mit einer Zweigstelle verbunden. Empfängt dieser Router nun ein Datagramm von irgendeiner dieser Verbindungen, so wird über das Routing festgelegt, über welche der Verbindungen das Datagramm weitergereicht wird. Jeder Rechner benötigt das Routing, denn selbst der einfachste Rechner am Internet besitzt mindestens zwei Netzwerk Schnittstellen, nämlich das Loopback Interface sowie die normale Schnittstelle zum restlichen Netzwerk, also Ethernet, PPP oder SLIP.

Also, wie funktioniert nun dieses Routing? Jeder einzelne Rechner hat eine eigene Liste mit Vorschriften für das Routing, man nennt sie die *Routing Table*. Diese Tabelle enthält Zeilen mit typischerweise mindestens drei Spalten. Die erste Spalte enthält die Zieladresse, die zweite Spalte die Schnittstelle, über die das Datenpaket weitergeleitet werden soll, und die dritte enthält optional die IP Adresse eines anderen Rechners (Gateway), der das Datenpaket zu seinem nächsten Etappenziel leitet. Unter Linux kann man sich diese Tabelle mit dem folgenden Befehl ansehen:



```
# cat /proc/net/route
```

Der eigentliche Vorgang des Routing ist sehr einfach: Ein eingehendes Datenpaket wird entgegengenommen, seine Zieladresse wird untersucht und mit den Einträgen in der Tabelle verglichen. Der Eintrag, der der Zieladresse am besten entspricht, wird selektiert und das Datenpaket an die in diesem Eintrag festgelegte Schnittstelle weitergeleitet. Ist für die Adresse auch ein Gateway eingetragen, wird das Paket an diesen Rechner adressiert, andernfalls wird angenommen, daß der Zielrechner zu dem Netzwerk gehört, mit dem die benutzte Schnittstelle verbunden ist.

Um die Routing Table zu verändern gibt es einen speziellen Befehl. Dieser Befehl übernimmt die Kommandozeilenargumente und setzt sie in die entsprechenden Systemaufrufe um die den Kernel dazu veranlassen, die entsprechenden Einträge in der Routing Table hinzuzufügen, zu entfernen oder zu verändern. Aus naheliegenden Gründen heißt dieser Befehl `route`.

Ein einfaches Beispiel: Nehmen wir an wir befinden uns an einem Ethernet Netzwerk. Es sei ein Klasse C Netz mit der Adresse `192.168.1.0`. Unsere eigene IP Adresse ist `192.168.1.10`, und der Rechner `192.168.1.1` ist ein Router mit Verbindung zum Internet.

Zunächst muß natürlich die Schnittstelle wie bereits beschrieben konfiguriert werden, also etwa

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Als nächstes muß in die Routing Table eingetragen werden, daß Datagramme an alle Adressen `192.168.1.*` direkt über das Ethernet Device `eth0` geleitet werden:

```
# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

Über den Schalter `-net` im obigen Befehl wird dem `route` Programm mitgeteilt, daß es sich um einen Eintrag für ein ganzes Netzwerk handelt. Die andere Alternative ist ein Eintrag `host`, bei dem nur eine einzige IP Adresse eingegeben wird.

Mittels diesem Eintrag ist der eigene Rechner nun in der Lage, zu allen anderen Rechnern im lokalen Ethernet IP Verbindungen aufzubauen. Doch was ist mit Rechnern, die sich außerhalb dieses Netzes befinden?

Es wäre sehr umständlich und nicht praktikabel, für jedes denkbare Netzwerk einen entsprechenden Eintrag anzufügen. Aus diesem Grund gibt es eine Standardeinstellung in der festgelegt wird, wie mit Paketen zu verfahren ist, die nicht gesondert in der Routing Table aufgeführt sind: die *Default Route*. Im obigen Beispiel heißt das: Alles was nicht im lokalen Netz ist, wird über den Router weitergeleitet - der wird dann schon wissen, wie mit dem Paket zu verfahren ist. Den entsprechenden Eintrag in der Routing Table erzeugt man folgendermaßen:

```
# route add default gw 192.168.1.1 eth0
```

Durch den Parameter `gw` wird dem `route`-Befehl mitgeteilt daß die folgende Adresse die IP-Adresse eines Gateway Rechners oder eines Routers ist, an den die Pakete weitergeleitet werden.

Die komplette Konfiguration sieht also so aus:

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
# route add default gw 192.168.1.1 eth0
```

Ein Blick in die `rc`-Dateien, die beim Bootprozeß das Netzwerk initialisieren, sollte ähnliche Einträge (wenn auch mit anderen Adressen) zu Tage bringen, denn es ist eine sehr verbreitete Konfiguration.

Wir können uns nun an ein etwas komplizierteres Beispiel wagen. Nehmen wir an, wir wollten einen einfachen Router konfigurieren, z.B. den bereits erwähnten mit mehreren lokalen Netzen und einer PPP Verbindung zum Internet. Für drei lokale Ethernet Segmente würde die Routing Table etwa folgendermaßen aufgebaut:

```
# route add 192.168.1.0 netmask 255.255.255.0 eth0
# route add 192.168.2.0 netmask 255.255.255.0 eth1
# route add 192.168.3.0 netmask 255.255.255.0 eth2
# route add default ppp0
```

Für jede der an diesen Router angeschlossenen Workstations hätte die Routing Table dieselbe einfache Form wie im vorangegangenen Beispiel, lediglich der Router muß alle drei Netzwerke separat auflisten, da er ja die Aufteilung der Datenpakete auf diese Netze durchführen muß. Bleibt also nur noch die Frage, warum in der default route der Eintrag `gw` fehlt. Der Grund dafür ist, daß es sich bei einer PPP-Verbindung (ebenso wie bei einer Verbindung über das SLIP Protokoll) um eine Verbindung zwischen genau zwei Rechnern handelt. Der Kernel 'weiß' also welchen Rechner er über die PPP-Verbindung anspricht, und die zusätzliche Angabe einer Gateway-Adresse ist in diesem Falle überflüssig. Lediglich für Ethernet-, Arcnet- oder Token Ring Verbindungen ist die Angabe einer Gatewayadresse zwingend vorgeschrieben, da hier über eine Verbindung eine große Zahl an Rechnern erreicht werden kann.

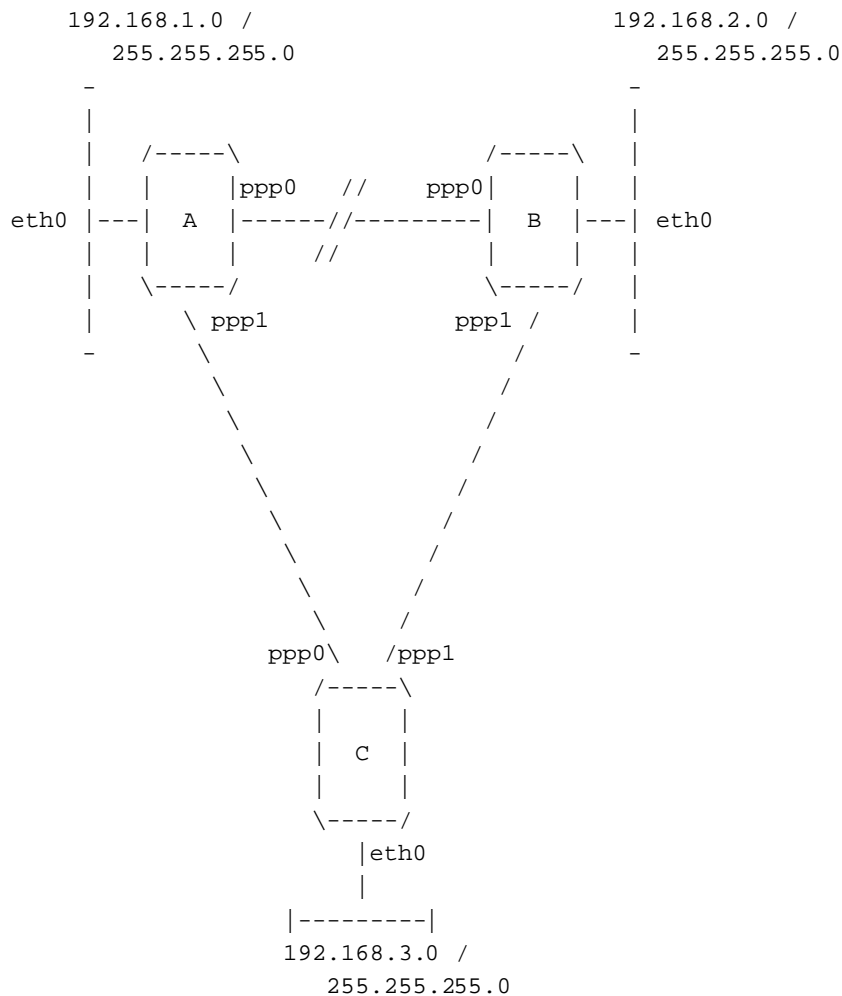
#### 4.7.1 Was macht das `routed` Programm?

Die oben beschriebene Konfiguration ist optimiert auf einfache Netzwerke mit nur wenigen, unveränderlichen Pfaden zu den unterschiedlichen Zielen. In einem komplexen Netzwerk werden die Dinge jedoch etwas schwieriger. Doch zum Glück betrifft das nur die wenigsten.

Das größte Problem des manuellen oder statischen Routing, das im vorigen Abschnitt beschrieben wurde, tritt auf, wenn ein (Verbindungs-) Rechner im Netzwerk ausfällt. In diesem Fall besteht die einzige Möglichkeit, ein Datenpaket dennoch zum Ziel weiterzuleiten darin, von Hand einzugreifen und die entsprechenden Routes manuell einzutragen – vorausgesetzt natürlich, es existiert solch ein alternativer Weg. Das ist umständlich, langsam und fehleranfällig. Deshalb wurden unterschiedliche Mechanismen entwickelt, um die Routing Table automatisch anzupassen, falls ein Netzwerkfehler auftritt und 'Umwege' zum Ziel bekannt sind. All diese Techniken bezeichnet man als *dynamische Routing Protokolle*.

Die bekanntesten dynamischen Protokolle sind RIP (Routing Information Protocol) und OSPF (Open Shortest Path First Protocol). RIP ist besonders in kleinen Netzwerken wie mittelgroßen Betrieben oder Gebäude-Netzwerken sehr verbreitet. OSPF ist moderner und insbesondere darauf ausgelegt, in großen Netzwerken benutzt zu werden, in denen es eine große Zahl an Wegen durch das Netzwerk gibt. Die am weitesten verbreiteten Vertreter dieser Protokolle sind `routed` (RIP) und `gated` (OSPF). `routed` ist normalerweise Bestandteil jeder Linux Distribution, sonst bekommt man es mit dem Paket NetKit (s.o.).

Ein Beispiel für die Verwendung dynamischen Routings ist die folgende Konfiguration:



Es gibt drei Router: A, B und C. Jeder ist für ein Ethernet Segment mit einem Klasse C IP Netzwerk (netmask 255.255.255.0) zuständig. Darüberhinaus verfügt jeder Router über jeweils eine PPP-Verbindung zu jedem der anderen beiden Router, diese bilden also ein Dreieck.

Ganz offensichtlich könnte die Routing Table für Router A folgendermaßen aussehen:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Dies funktioniert aber nur, solange die Verbindung zwischen Router A und B besteht. Bricht diese Verbindung zusammen, können Rechner des Ethernet Segmentes von Router A keinen Rechner des Segmentes von Router B mehr erreichen, da A die Datagramme über die PPP-Verbindung an B weiterreichen will. Sie können immernoch Verbindungen zu den Rechnern des Segmentes C aufbauen, und diese wiederum können problemlos mit Rechnern im Segment B kommunizieren, da die Verbindung zwischen B und C immernoch besteht.

Es wäre also naheliegend daß A die an B gerichteten Pakete an C sendet und diese dann von C an B weitergeleitet werden. Für genau diese Art von Problemen sind die dynamischen Protokolle wie RIP ausgelegt. Würde auf jedem der drei Router A, B und C ein Routing Dämon laufen, würden diese im Falle eines Netzwerkfehlers die Routing Tabellen der drei Router automatisch den neuen Gegebenheiten anpassen. Ein derartiges Netz zu konfigurieren ist sehr einfach, es sind lediglich zwei Schritte notwendig. Hier das Beispiel für Router A:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# /usr/sbin/routed
```

Der Routing Dämon `routed` erkennt beim Start automatisch alle aktiven Netzwerkschnittstellen und sendet und erkennt über diese Schnittstellen Meldungen um festzustellen, ob Änderungen in der Routing Table nötig sind.

Die war nur eine sehr kurze Beschreibung, was dynamisches Routing ist, und in welchen Fällen man es verwendet. Wer genauere Informationen benötigt sei auf die am Anfang dieses Textes genannten Quellen verwiesen.

Wichtige Punkte im Zusammenhang mit dynamischen Routing sind:

1. Einen Routing Dämon benötigt nur, wer auf seinem Rechner mehrere verschiedene mögliche Routes zu einer Zieladresse besitzt.
2. Der Routing Dämon ändert automatisch die Routing Table, um sie an Änderungen im Netzwerk anzupassen.
3. RIP ist für kleine und mittelgroße Netzwerke ausgelegt.

## 4.8 Die Konfiguration von Netzwerk Servern und Diensten.

Netzwerk Server und Dienste bezeichnet diejenigen Programme, die es einem Nutzer von außerhalb (remote user) erlauben, ihren Rechner zu benutzen. Dieser Nutzer stellt eine Netzwerkverbindung zu ihrem Rechner, oder besser zu einem Server-Programm auf ihrem Rechner, her. Dieser Server, man nennt ihn auch Netzwerk Dämon, überwacht einen *Port*. Er nimmt ankommende Verbindungswünsche entgegen und führt dann die jeweiligen Aktionen aus. Es gibt zwei unterschiedliche Methoden, wie ein solcher Netzwerk-Dämon arbeitet:

### Standalone

Der Dämon überwacht selber den Port. Im Falle einer ankommenden Verbindung übernimmt der Dämon selbst die Arbeit und stellt die gewünschte Dienstleistung zur Verfügung.

### slave des `inetd` Servers

Der `inetd` Server ist ein besonderer Dämon der allgemein darauf spezialisiert ist, eingehende Netzwerkverbindungen zu beantworten. Er besitzt eine eigene Konfigurationsdatei in der festgelegt wird, welche Programme er starten muß wenn auf einem Port eine tcp oder udp Anfrage eintrifft. Diese Ports werden in einer anderen Datei beschrieben, davon später mehr.

Es gibt zwei wichtige Konfigurationsdateien, die an die eigenen Bedürfnisse angepaßt werden müssen. Dies sind `/etc/services`, in dem den unterschiedlichen Portnummern Namen zugeordnet werden, und `/etc/inetd.conf`, die Konfigurationsdatei des `inetd` Netzwerkdamons.

### 4.8.1 `/etc/services`

Die Datei `/etc/services` ist eine einfache Datenbasis, die jedem Port einen für Menschen leichter verständlichen Namen zuordnet. Das Format dieser Datei ist sehr einfach: Es handelt sich um eine Textdatei, und jede Zeile stellt einen Eintrag der Datenbasis dar. ein solcher Eintrag besteht aus drei Feldern, durch beliebig viele Leerzeichen getrennt. Diese drei Felder sind:

Name	Port/Protokoll	Aliases	# Kommentar
------	----------------	---------	-------------

#### Name

Ein einzelnes Wort, welches den jeweiligen Service beschreibt.

#### Port/Protokoll

Dieses Feld besteht aus zwei Einträgen.

#### Port

Eine Nummer, die die Portnummer angibt, unter der der jeweilige Service angesprochen werden kann. Die meisten der üblichen Services haben festgelegte Nummern, dies wird in RFC-1340 beschrieben.

### Protokoll

Je nach verwendetem Protokoll steht hier `tcp` oder `udp`

Es ist wichtig darauf hinzuweisen daß ein Eintrag `18/tcp` etwas ganz anderes ist als ein Eintrag `18/udp`. Es gibt keinen technischen Grund warum ein Service über beide Protokolle zur Verfügung stehen sollte. Nur in seltenen Ausnahmefällen ist dies der Fall, dann wird man beide Einträge, also für `udp` und `tcp` finden.

### Aliases

Zusätzliche Namen, unter denen dieser Service angesprochen werden kann.

Jeglicher Text nach dem Hash-Zeichen (`#`) wird ignoriert.

**Ein Beispiel für `/etc/services`.** Alle modernen Linux Distributionen enthalten bereits eine gute Version dieser Datei. Falls aber jemand seinen eigenen Rechner von Grund auf selber aufbauen will, hier ist die mit der Debian Distribution gelieferte Version.

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, ``Assigned Numbers'' (July 1992). Not all ports
# are included, only the more common ones.

tcpmux      1/tcp                # TCP port service multiplexer
echo        7/tcp
echo        7/udp
discard     9/tcp                sink null
discard     9/udp                sink null
sysstat     11/tcp               users
daytime     13/tcp
daytime     13/udp
netstat     15/tcp
qotd        17/tcp               quote
msp         18/tcp               # message send protocol
msp         18/udp               # message send protocol
chargen     19/tcp               ttytst source
chargen     19/udp               ttytst source
ftp-data    20/tcp
ftp         21/tcp
ssh         22/tcp               # SSH Remote Login Protocol
ssh         22/udp               # SSH Remote Login Protocol
telnet      23/tcp
# 24 - private
smtp        25/tcp               mail
# 26 - unassigned
time        37/tcp               timserver
time        37/udp               timserver
rlp         39/udp               resource                # resource location
nameserver  42/tcp               name                    # IEN 116
whois       43/tcp               nickname
re-mail-ck  50/tcp               # Remote Mail Checking Protocol
re-mail-ck  50/udp               # Remote Mail Checking Protocol
```

```

domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp          # deprecated
bootps         67/tcp          # BOOTP server
bootps         67/udp
bootpc         68/tcp          # BOOTP client
bootpc         68/udp
tftp           69/udp
gopher         70/tcp          # Internet Gopher
gopher         70/udp
rje            77/tcp          netrjs
finger        79/tcp
www           80/tcp          http            # WorldWideWeb HTTP
www           80/udp          # HyperText Transfer Protocol
link          87/tcp          ttylink
kerberos      88/tcp          kerberos5 krb5 # Kerberos v5
kerberos      88/udp          kerberos5 krb5 # Kerberos v5
supdup        95/tcp
# 100 - reserved
hostnames     101/tcp        hostname        # usually from sri-nic
iso-tsap      102/tcp        tsap            # part of ISODE.
csnet-ns     105/tcp        cso-ns          # also used by CSO name server
csnet-ns     105/udp        cso-ns
rtelnet      107/tcp        # Remote Telnet
rtelnet      107/udp
pop-2        109/tcp        postoffice      # POP version 2
pop-2        109/udp
pop-3        110/tcp        # POP version 3
pop-3        110/udp
sunrpc       111/tcp        portmapper      # RPC 4.0 portmapper TCP
sunrpc       111/udp        portmapper      # RPC 4.0 portmapper UDP
auth         113/tcp        authentication tap ident
sftp         115/tcp
uucp-path    117/tcp
nntp         119/tcp        readnews untp   # USENET News Transfer Protocol
ntp          123/tcp        # Network Time Protocol
ntp          123/udp        # NETBIOS Name Service
netbios-ns   137/tcp        # NETBIOS Datagram Service
netbios-ns   137/udp
netbios-dgm  138/tcp        # NETBIOS session service
netbios-dgm  138/udp
netbios-ssn  139/tcp        # Interim Mail Access Proto v2
netbios-ssn  139/udp
imap2        143/tcp        # Simple Net Mgmt Proto
imap2        143/udp        # Traps for SNMP
snmp         161/udp        snmptrap        # ISO mgmt over IP (CMOT)
snmp-trap    162/udp
cmip-man     163/tcp
cmip-man     163/udp
cmip-agent   164/tcp
cmip-agent   164/udp
xdmcp        177/tcp        # X Display Mgr. Control Proto
xdmcp        177/udp
nextstep     178/tcp        NeXTStep NextStep # NeXTStep window
nextstep     178/udp        NeXTStep NextStep # server
bgp          179/tcp        # Border Gateway Proto.
bgp          179/udp

```

```

prospero      191/tcp      # Cliff Neuman's Prospero
prospero      191/udp
irc           194/tcp      # Internet Relay Chat
irc           194/udp
smux         199/tcp      # SNMP Unix Multiplexer
smux         199/udp
at-rtmp      201/tcp      # AppleTalk routing
at-rtmp      201/udp
at-nbp       202/tcp      # AppleTalk name binding
at-nbp       202/udp
at-echo      204/tcp      # AppleTalk echo
at-echo      204/udp
at-zis       206/tcp      # AppleTalk zone information
at-zis       206/udp
z3950        210/tcp      wais         # NISO Z39.50 database
z3950        210/udp      wais
ipx          213/tcp      # IPX
ipx          213/udp
imap3        220/tcp      # Interactive Mail Access
imap3        220/udp      # Protocol v3
ulistserv    372/tcp      # UNIX Listserv
ulistserv    372/udp
#
# UNIX specific services
#
exec         512/tcp
biff         512/udp      comsat
login        513/tcp
who          513/udp      whod
shell        514/tcp      cmd          # no passwords used
syslog       514/udp
printer      515/tcp      spooler      # line printer spooler
talk         517/udp
ntalk        518/udp
route        520/udp      router routed # RIP
timed        525/udp      timeserver
tempo        526/tcp      newdate
courier      530/tcp      rpc
conference   531/tcp      chat
netnews      532/tcp      readnews
netwall      533/udp      # -for emergency broadcasts
uucp         540/tcp      uucpd        # uucp daemon
remotefs     556/tcp      rfs_server rfs # Brunhoff remote filesystem
klogin       543/tcp      # Kerberized 'rlogin' (v5)
kshell       544/tcp      krcmd        # Kerberized 'rsh' (v5)
kerberos-adm 749/tcp      # Kerberos 'kadmin' (v5)
#
webster       765/tcp      # Network dictionary
webster       765/udp
#
# From ``Assigned Numbers``:
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical

```

```

#> connections which carry long term conversations.  For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined.  This list specifies the port used by the server process as its
#> contact port.  While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convenience to the
#> community.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np    1525/tcp          # Prospero non-privileged
prospero-np    1525/udp
rfe            5002/tcp          # Radio Free Ethernet
rfe            5002/udp          # Actually uses UDP only
bbs            7000/tcp          # BBS service
#
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4, and are unofficial.  Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
kerberos4      750/udp          kdc      # Kerberos (server) udp
kerberos4      750/tcp          kdc      # Kerberos (server) tcp
kerberos_master 751/udp          # Kerberos authentication
kerberos_master 751/tcp          # Kerberos authentication
passwd_server  752/udp          # Kerberos passwd server
krb_prop       754/tcp          # Kerberos slave propagation
krbupdate      760/tcp          kreg     # Kerberos registration
kpasswd        761/tcp          kpwd     # Kerberos "passwd"
kpop           1109/tcp         # Pop with Kerberos
knetd          2053/tcp         # Kerberos de-multiplexor
zephyr-srv    2102/udp         # Zephyr server
zephyr-clt    2103/udp         # Zephyr serv-hm connection
zephyr-hm     2104/udp         # Zephyr hostmanager
eklogin       2105/tcp         # Kerberos encrypted rlogin
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv    871/tcp          # SUP server
supfiledbg    1127/tcp         # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp          1/ddp          # Routing Table Maintenance Protocol
nbp           2/ddp          # Name Binding Protocol
echo         4/ddp          # AppleTalk Echo Protocol
zip          6/ddp          # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg       1236/tcp         # Gracilis Packeten remote config server
xtel         1313/tcp         # french minitel
cfinger      2003/tcp         # GNU Finger
postgres     4321/tcp         # POSTGRES
mandelspawn  9359/udp         mandelbrot # network mandelbrot

# Local services

```



### 4.8.2 /etc/inetd.conf

Die Datei `/etc/inetd.conf` ist die Konfigurationsdatei des Server Dämons `inetd`. Bei einer eingehenden Anfrage nach einem bestimmten Service sieht der Dämon in dieser Datei nach, was zu tun ist. Für jeden Service, den man anbieten will, muß ein entsprechender Eintrag vorhanden sein in dem festgelegt wird, welcher Dämon bei einer Anfrage gestartet werden soll, und wie dies zu geschehen hat.

Auch hier ist das Dateiformat sehr einfach, es handelt sich ebenfalls um eine reine Textdatei, in der in jeder Zeile ein anzubietender Service beschrieben wird. Das Zeichen `#` dient als Kommentarzeichen, nachfolgender Text wird ignoriert. Jede Zeile enthält sieben Felder, die jeweils durch eine beliebige Anzahl von Leerzeichen oder Tabulatoren voneinander getrennt sind. Die Bezeichnungen der einzelnen Felder sind folgende:

```
service socket_type proto flags user server_path server_args
```

#### service

Name des Dienstes, entsprechend dem Eintrag in `etc/services`.

#### socket\_type

Dieser Eintrag beschreibt den Typ des Socket, der für den Dienst gilt. Erlaubte Einträge sind `stream`, `dgram`, `raw`, `rdm` und `seqpacket`. Die Gründe für die Unterteilung sind technischer Natur, aber als Faustregel kann man davon ausgehen, daß praktisch alle `tcp`-basierten Dienste `stream` verwenden, während `udp`-basierte Dienste `dgram` benutzen. Nur in ganz seltenen Fällen wird ein Dienst einen anderen Typ verwenden.

#### proto

Das für diesen Service gültige Protokoll. Es muß mit dem Eintrag in `/etc/services` übereinstimmen, normalerweise also entweder `tcp` oder `udp`. Für Server, die Sun RPC (Remote Procedure Call) verwenden, lauten die Einträge `rpc/tcp` oder `rpc/udp`.

#### flags

Hier gibt es nur zwei mögliche Einträge. Dem `inetd` Server wird damit angezeigt, ob das gestartete Serverprogramm den Socket nach dem Start wieder freigibt oder nicht. Danach entscheidet sich, ob für eine weitere eingehende Anfrage ein neuer Prozeß gestartet werden muß, oder ob der laufende Prozeß auch die neuen Anfragen bearbeitet. Die Regeln hierfür sind etwas schwierig, aber auch hier gilt als Faustregel: `tcp`-Dienste benötigen den Eintrag `nowait`, `udp`-Dienste verwenden `wait`. Es gibt hier aber etliche Ausnahmen, im Zweifelsfall sollte man sich am Beispiel orientieren.

#### user

Dieser Eintrag legt den Nutzernamen entsprechend `/etc/passwd` fest, mit dessen Rechten der Server gestartet wird. Dies wird oft aus Sicherheitsgründen gemacht. Verwendet man hier der Benutzer `nobody`, so werden die möglichen Folgeschäden eingegrenzt, sollte doch jemand die Sicherheitsmechanismen des Systems umgehen. Allerdings benötigen viele Server die Rechte des Systemadministrators, deshalb steht hier meist `root`.

#### server\_path

Dies ist der Name inklusive vollem Pfad des zu startenden Servers.

#### server\_args

Dieser Eintrag umfaßt die gesamte restliche Zeile und ist optional. Hier können zusätzliche Argumente für das Serverprogramm übergeben werden.

**Ein Beispiel für `/etc/inetd.conf`** Wie auch im Falle von `/etc/services` gehört ein funktionierendes `/etc/inetd.conf` zum Standardumfang jeder Distribution. Der Vollständigkeit halber hier die Version der Debian Distribution.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emen.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Internal services
#
#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
time          stream  tcp    nowait  root    internal
time          dgram  udp    wait    root    internal
#
# These are standard services.
#
telnet        stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp           stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp         dgram  udp    wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec and talk are BSD protocols.
#
shell         stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login         stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec        stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk          dgram  udp    wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk         dgram  udp    wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp          stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp        stream  tcp    nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp        stream  tcp    nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat      dgram  udp    wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et al
#
#pop-2       stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3       stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# 'cfinger' is for the GNU finger server available for Debian.  (NOTE: The
# current implementation of the 'finger' daemon allows it to be run as 'root'.)
#
#cfinger     stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger      stream  tcp    nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat     stream  tcp    nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat      stream  tcp    nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwx
#
# Tftp service is provided primarily for booting.  Most sites
```

```

# run this only on machines acting as "boot servers."
#
#tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd
#tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /boot
#bootps dgram udp wait root /usr/sbin/bootpd bootpd -i -t 120
#
# Kerberos authenticated services (these probably need to be corrected)
#
#klogin stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rlogind -k
#eklogin stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rlogind -k -x
#kshell stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be corrected)
#
#krbupdate stream tcp nowait root /usr/sbin/tcpd /usr/sbin/registerd
#kpasswd stream tcp nowait root /usr/sbin/tcpd /usr/sbin/kpasswd
#
# RPC based services
#
#mountd/1 dgram rpc/udp wait root /usr/sbin/tcpd /usr/sbin/rpc.mountd
#rstatd/1-3 dgram rpc/udp wait root /usr/sbin/tcpd /usr/sbin/rpc.rstatd
#rusersd/2-3 dgram rpc/udp wait root /usr/sbin/tcpd /usr/sbin/rpc.rusersd
#walld/1 dgram rpc/udp wait root /usr/sbin/tcpd /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident stream tcp nowait nobody /usr/sbin/identd identd -i

```

## 4.9 Weitere Konfigurationsdateien im Netzwerkkumfeld.

Es gibt noch eine ganze Reihe an Dateien, die mit der Netzwerkkonfiguration unter Linux zu tun haben. Die meisten davon wird man nie verändern müssen, es lohnt sich aber dennoch, sie kurz zu beschreiben, damit klar wird was darinsteht, und wozu sie gut sind.

### 4.9.1 /etc/protocols

/etc/protocols ist eine Datei, in der Protokollnamen und Identifikationsnummern einander zugeordnet werden. Sie wird vorwiegend von Programmierern verwendet, damit sie in ihren Programmen die Dienste anhand ihres Namens verwenden können, außerdem von Programmen wie `tcpdump`, um anstelle von Nummern Namen ausgeben zu können. Die Standardsyntax dieser Datei ist

```
Protokollname Nummer Alias
```

Die Datei /etc/protocols der Debian Distribution sieht so aus:

```

# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
# from: @(#)protocols 5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).

ip 0 IP # internet protocol, pseudo protocol number
icmp 1 ICMP # internet control message protocol

```

```

igmp      2      IGMP      # Internet Group Management
ggp       3      GGP       # gateway-gateway protocol
ipencap   4      IP-ENCAP  # IP encapsulated in IP (officially ``IP'')
st        5      ST        # ST datagram mode
tcp       6      TCP       # transmission control protocol
egp       8      EGP       # exterior gateway protocol
pup      12      PUP       # PARC universal packet protocol
udp      17      UDP       # user datagram protocol
hmp      20      HMP       # host monitoring protocol
xns-idp  22      XNS-IDP   # Xerox NS IDP
rdp      27      RDP       # "reliable datagram" protocol
iso-tp4  29      ISO-TP4   # ISO Transport Protocol class 4
xtp      36      XTP       # Xpress Transfer Protocol
ddp      37      DDP       # Datagram Delivery Protocol
idpr-cmt  39      IDPR-CMTP # IDPR Control Message Transport
rspf     73      RSPF      # Radio Shortest Path First.
vmtp     81      VMTP      # Versatile Message Transport
ospf     89      OSPFIGP   # Open Shortest Path First IGP
ipip     94      IPIP      # Yet Another IP encapsulation
encap    98      ENCAP     # Yet Another IP encapsulation

```

#### 4.9.2 /etc/networks

Die Datei `/etc/networks` hat eine ähnliche Funktion wie `/etc/hosts`. Es stellt eine einfache Datenbasis für die Zuordnung von Netzwerknamen und -adressen dar. Der einzige Unterschied zu letzterem besteht darin, daß nur zwei Einträge je Zeile erlaubt sind, und zwar folgendermassen:

```
Netzwerkname  Netzwerkadresse
```

Auch hier ein kleines Beispiel:

```

loopnet      127.0.0.0
localnet     192.168.0.0
amprnet      44.0.0.0

```

Bei Programmen wie `routew` wird ein Netzwerk, das einen Eintrag in `/etc/networks` hat, mit seinem Namen anstelle der reinen Netzwerkadresse angezeigt.

## 4.10 Netzwerksicherheit und Zugangskontrolle

Zu Beginn dieses Abschnittes eine kleine Warnung: Einen Rechner oder gar ein Netzwerk gegen unerlaubtes Eindringen abzusichern ist ein äußerst schwieriges Unterfangen. Ich selber betrachte mich nicht als Experten auf diesem Gebiet, und obwohl die im folgenden beschriebenen Mechanismen sicherlich hilfreich sind möchte ich all denen, die wirklich um die Sicherheit ihres Systems besorgt sind, raten, selber geeignete Literatur zu suchen. Im Internet findet man viele gute Hinweise dazu.

Ein wichtiger Grundsatz zur Sicherheit ist **Aktivieren sie keine Dienste, die sie nicht benötigen**. Die meisten Distributionen sind heute mit einer Vielzahl von Servern ausgestattet, die beim Bootprozeß automatisch gestartet werden. Um ein Mindestmaß an Systemsicherheit zu gewährleisten sollten Sie sich die Datei `/etc/inetd.conf` in Ruhe ansehen und alle nichtbenötigten Dienste durch Einfügen eines `#` am Zeilenanfang auszukommentieren. Gute 'Kandidaten' hierfür sind `shell`, `login`, `exec`, `uucp` und `ftp` sowie informelle Dienste wie `finger`, `netstat` und `sysstat`.

Es gibt eine große Zahl an Sicherheits- und Zugangskontrollmechanismen, ich werde im folgenden die wichtigsten davon kurz beschreiben.

#### 4.10.1 /etc/ftpusers

Die Datei `/etc/ftpusers` bietet eine einfache Möglichkeit, einzelne Personen vom Zugang über `ftp` auszuschließen. Die Datei wird vom `ftp`-Dämonen `ftpd` gelesen, wenn eine FTP-Verbindung aufgebaut wird. Die Datei enthält einfach eine Liste mit den Benutzernamen all derer, denen ein Login verboten werden soll. Hier ein Beispiel:

```
# /etc/ftpusers - users not allowed to login via ftp
root
uucp
bin
mail
```

#### 4.10.2 /etc/securetty

Mit dieser Datei wird festgelegt, an welchen (virtuellen) Terminals (`tty`s) sich der Systemverwalter `root` einloggen darf. `/etc/securetty` wird vom Login-Programm (normalerweise `/bin/login`) gelesen und enthält eine Liste der erlaubten Terminals, auf allen anderen kann `root` sich nicht einloggen:

```
# /etc/securetty - tty's on which root is allowed to login
tty1
tty2
tty3
tty4
```

#### 4.10.3 Die `tcpd` Hostzugangskontrolle

Das Programm `tcpd` ist ihnen vielleicht schon in der Datei `/etc/inetd.conf` aufgefallen. Es stellt Kontroll- und Zugangskontrollmechanismen für diejenigen Dienste zur Verfügung, für die es konfiguriert wird.

Wird es von `inetd` gestartet so liest es zwei Dateien, anhand derer der Zugang zum überwachten Server gewährt oder verboten werden kann.

Die beiden Steuerdateien werden jeweils solange gelesen, bis ein zutreffender Eintrag gefunden wird. Wird ein solcher zutreffender Eintrag nicht gefunden, wird angenommen, daß der Zugang für jeden erlaubt ist. Gelesen werden die Dateien in der Reihenfolge `/etc/hosts.allow`, `/etc/hosts.deny`. Die beiden Dateien werden in den folgenden Abschnitten beschrieben. Für eine detaillierte Beschreibung sei auf die man Hilfeseiten verwiesen, `host_access(5)` ist hier ein guter Startpunkt.

**`/etc/hosts.allow`** Dies ist eine der Konfigurationsdateien des Programmes `/usr/sbin/tcpd`. In `/etc/hosts.allow` wird eingestellt, welchen anderen Rechnern der Zugang zu Diensten auf dem eigenen Rechner gestattet werden soll. Das Dateiformat ist sehr einfach:

```
# /etc/hosts.allow
#
# <service list>: <host list> [: command]
```

##### **service list**

Eine durch Kommata getrennte Liste von Namen der Dienste, für die der Eintrag gelten soll, z.B. `ftpd`, `telnetd` oder `fingerd`.

##### **host list**

Eine durch Komma getrennte Liste von Rechnernamen, es können hier auch IP-Adressen angegeben werden. Außerdem können Platzhalter verwendet werden. Beispiele hierfür sind `gw.vk2ktj.ampr.org` (bestimmter Rechner), `.uts.edu.au` (alle Rechner deren Name mit dieser Zeichenkette endet) oder `44.` (alle IP-Adressen, die mit der angegebenen Ziffernfolge beginnen). Weiterhin existieren einige besondere, die die Konfiguration vereinfachen. Einige davon sind `ALL` (jeder Rechner), `LOCAL` (Rechner ohne Dezimalpunkt `."` im Namen, also solche der lokalen Domain) sowie `PARANOID` (Rechner, deren Namen nicht der Adresse entspricht; dient der Vermeidung von Spoofing). Ein letzter nützlicher Eintrag ist `EXCEPT`. Dadurch können Listen mit Ausnahmen definiert werden, wie in einem späteren Beispiel erläutert wird.

#### command

Dies ist ein optionaler Parameter. Hier kann ein Programm mit seinem vollständigen Pfad angegeben werden welches jedesmal ausgeführt wird, wenn die entsprechende Regel erfüllt ist. Es kann beispielsweise ein Programm gestartet werden das herauszufinden versucht, wer gerade auf dem anderen Rechner eingelogged ist, oder eine Meldung an den Systemadministrator schickt, daß gerade jemand versucht, diesen Dienst zu nutzen. Zur Kommandogenerierung existieren einige Platzhalter, die automatisch gesetzt werden: `%h` ist der Name des Rechners, der die Verbindung aufbauen will, oder seine IP-Adresse. `%d` ist der Name des Dämons, der gestartet werden soll.

Ein Beispiel:

```
# /etc/hosts.allow
#
# Mail ist jedem erlaubt
in.smtpd: ALL
# Telnet und FTP nur von lokalen Rechnern sowie meinem Rechner zu Hause
telnetd, ftpd: LOCAL, myhost.athome.org.au
# Finger ist allen erlaubt, aber es wird protokolliert, woher die
# Anfrage kommt
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

**/etc/hosts.deny** Dies ist eine der Konfigurationsdateien des Programmes `/usr/sbin/tcpd`. In `/etc/hosts.deny` wird eingestellt, welchen anderen Rechnern der Zugang zu Diensten auf dem eigenen Rechner verboten werden soll.

Ein einfaches Beispiel sieht etwa so aus:

```
# /etc/hosts.deny
#
# Kein Zugang fuer Rechner mit suspektem Namen
ALL: PARANOID
#
# Verbot fuer ALLE Rechner
ALL: ALL
```

Der Eintrag `PARANOID` ist hier redundant, da der folgende Eintrag in jedem Fall einen Zugang unterbindet. Jeder der beiden Einträge ist eine sinnvolle Einstellung, abhängig von den jeweiligen Bedürfnissen.

Die sicherste Konfiguration ist ein Eintrag `ALL: ALL` in `/etc/hosts.deny` zusammen mit einer Datei `/etc/hosts.allow` in der im einzelnen festgelegt wird, für wen der Zugang erlaubt wird.

#### 4.10.4 /etc/hosts.equiv

Die Datei `/etc/hosts.equiv` erlaubt es, einzelnen Rechnern und Benutzern den Zugang zur eigenen Maschine ohne Paßwortabfrage zu ermöglichen. Dies ist in einer sicheren Umgebung hilfreich, in der man alle anderen Maschinen unter Kontrolle hat. Andernfalls ist es aber ein großes Sicherheitsrisiko. Denn der eigene Rechner ist nur

so sicher wie der unsicherste Rechner, dem man vertraut. Wer großen Wert auf höchste Sicherheit legt sollte diesen Mechanismus nicht verwenden, und auch den Nutzern nahelegen, die Datei `.rhosts` nicht zu verwenden.

#### 4.10.5 Konfiguration des `ftp` Dämons

Viele Besitzer von vernetzten Rechnern sind daran interessiert, anderen Personen das Übertragen von Daten von und zum eigenen Rechner zu ermöglichen, ohne ihnen einen expliziten Account einzurichten. Dazu dient der FTP Server. Es muß aber sichergestellt sein, daß der `ftp`-Dämon korrekt für den anonymen Zugang konfiguriert ist. Die Seite `ftpd(8)` der Online-Hilfe beschreibt die dazu notwendigen Schritte in einiger Länge. Diesen Tips sollte man unbedingt folgen. Außerdem ein wichtiger Tip: Verwenden sie auf keinen Fall einfach eine Kopie der eigenen Datei `/etc/passwd` im anonymen Heimatverzeichnis `/etc`. Stellen sie sicher, daß alle unwichtigen Einträge entfernt werden, sonst stehen Angriffen durch Paßwortentschlüsselung Tür und Tor offen.

#### 4.10.6 Einrichtung eines Firewall

Eine extrem sichere Methode gegen Angriffe über das Netzwerk ist es, erst gar keine Datagramme an den Rechner heranzulassen. Dieses wird in einem eigenen Dokument beschrieben, dem *Firewall HOWTO*.

#### 4.10.7 Weitere Tips und Vorschläge

Hier noch ein paar weitere Hinweise, auch wenn der eine oder andere davon geeignet ist, Glaubenskriege unter Unix-Administratoren hervorzurufen.

##### **sendmail**

Obwohl die Verwendung des `sendmail`-Dämons sehr weit verbreitet ist, taucht er mit erschreckender Regelmäßigkeit in Warnungen vor Sicherheitslöchern auf. Es obliegt jedem selber, ob er `sendmail` verwenden will.

##### **NFS und andere Sun RPC Dienste**

Seien Sie vorsichtig damit. Es gibt bei diesen Diensten eine große Zahl potentieller Sicherheitsrisiken. Allerdings ist es schwierig, für etwas wie NFS eine Alternative zu finden. Wenn Sie diese Dienste benutzen, seien Sie vorsichtig, wem Sie einen Zugriff darauf erlauben.

## 5 Spezifische Informationen zur Netzwerk Technologie.

Die Informationen in den folgenden Abschnitten sind jeweils spezifisch für die jeweilige Technologie. Die darin gemachten Aussagen gelten nicht automatisch auch für andere Netzwerk Technologien.

### 5.1 ARCNet

Die Device Namen für ARCNET sind `arc0s`, `arc1e`, `arc2e` usw. Der ersten gefundenen Karte wird automatisch der Eintrag `arc0` zugewiesen, den weiteren Karten die folgenden Nummern in der Reihenfolge ihrer Detektion. Der Buchstabe am Ende des Devicenamens gibt an, ob als Paketformat Ethernet Encapsulation oder RFC1051 ausgewählt wurde.

#### **Optionen beim Kernel kompilieren:**

```
Network device support --->
  [*] Network device support
  <*> ARCnet support
```

```
[ ] Enable arc0e (ARCnet "Ether-Encap" packet format)
[ ] Enable arc0s (ARCnet RFC1051 packet format)
```

Ist die Unterstützung für die Karte erst einmal im Kernel eingebunden, ist die Konfiguration einfach. Typischerweise geschieht das etwa so:

```
# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
# route add 192.168.0.0 netmask 255.255.255.0 arc0e
```

Die Datei `/usr/src/linux/Documentation/networking/arcnet-hardware.txt` enthält weitere Informationen zu diesem Thema.

Die ARCNet Unterstützung wurde von Avery Pennarun (`apenwarr@foxnet.net`) entwickelt.

## 5.2 Appletalk (AF\_APPLETALK)

Hierfür gibt es keine speziellen Device-Einträge, da bestehende Netzwerk-Devices genutzt werden.

### Optionen beim Kernel kompilieren:

```
Networking options --->
<*> Appletalk DDP
```

Durch die Unterstützung von Appletalk kann ein Linux Rechner mit einem Apple Netzwerk zusammenarbeiten. Eine wichtige Anwendung dafür ist die gemeinsame Nutzung von Druckern oder Festplatten über ein Netzwerk. Man benötigt dafür zusätzliche Software: `netatalk`. Wesley Craig (`netatalk@umich.edu`) steht stellvertretend für ein Team an der University of Michigan, das sich 'Research Systems Unix Group' nennt. Sie haben das Paket `netatalk` mit der notwendigen Software entwickelt, nämlich der Implementation des Appletalk Protocoll Stack sowie weitere nützliche Hilfsprogramme. Das Paket `netatalk` ist entweder bereits Bestandteil ihrer Linux Distribution oder kann über ftp von der University of Michigan bezogen werden

```
terminator.rs.itd.umich.edu:/unix/netatalk/
```

Um das Paket zu übersetzen und zu installieren geht man folgendermassen vor:

```
# cd /usr/src
# tar xvfz ../netatalk-1.4b2.tar.Z
- Hier sollte man die Datei 'Makefile' editieren, um die Software an das
  eigene System anzupassen, z.B. die Variable DESTDIR, welche festlegt,
  wo die Dateien installiert werden.
# make
- als root:
# make install
```

### 5.2.1 Die Konfiguration der Appletalk Software.

Damit später alles einwandfrei funktioniert, sind zunächst einige zusätzliche Einträge in der Datei `/etc/services` nötig. Diese sind:

```
rtmp    1/ddp    # Routing Table Maintenance Protocol
nbp     2/ddp    # Name Binding Protocol
echo    4/ddp    # AppleTalk Echo Protocol
zip     6/ddp    # Zone Information Protocol
```



Als nächstes müssen die Konfigurationsdateien im Verzeichnis `/usr/local/atalk/etc` angelegt werden (oder wo immer das Paket installiert wurde).

Die erste Datei ist `atalkd.conf`, man benötigt hier vorläufig nur eine einzige Zeile in der festgelegt wird, über welches Netzwerk Device die Apple Rechner erreicht werden:

```
eth0
```

Der Appletalk Dämon wird nach seinem Start weitere Details hinzufügen.

### 5.2.2 Exportieren eines Linux Dateisystems via Appletalk,

Man kann Dateisysteme des Linuxrechners auch an Apple-Rechner exportieren, sodaß diese von beiden Rechnern gemeinsam genutzt werden können.

Dafür muß man die Datei `/usr/local/atalk/etc/AppleVolumes.system` entsprechend konfigurieren. Im selben Verzeichnis gibt es außerdem noch die Datei `AppleVolumes.default`. Sie hat dasselbe Format und legt fest, welche Dateisysteme für Nutzer zur Verfügung stehen, die sich als Gastnutzer anmelden.

Die genauen Details für diese Konfiguration entnehmen sie bitte der Manpage `afpd`. Eine einfache Konfiguration könnte etwa so aussehen:

```
/tmp Scratch  
/home/ftp/pub "Public Area"
```

Dadurch wird das lokale Verzeichnis `/tmp` als AppleShare Volume 'Scratch' und das öffentliche ftp-Verzeichnis als AppleShare Volume 'Public Area' exportiert. Die Namen für die Volumes müssen nicht angegeben werden. Wenn sie fehlen, weist der Dämon automatisch passende Namen zu.

### 5.2.3 Gemeinsame Nutzung eines Druckers mit Appletalk.

Die gemeinsame Nutzung eines Druckers läßt sich einfach verwirklichen. Man muß dazu das Programm `papd` starten, den Appletalk Printer Access Protocol Dämon. Dieses Programm übernimmt die Druckaufträge von Applerechnern im Netz und leitet sie an den lokale Drucker Spool Dämon weiter.

Zur Konfiguration dieses Dämon dient die Datei `papd.conf`. Die Syntax entspricht dabei der der Datei `/etc/printcap`. Der Name, der in der Datei definiert wird, wird dann über das Appletalk Naming Protokoll, NBP, registriert.

Hier eine Beispielkonfiguration:

```
TricWriter:\  
:pr=lp:op=cg:
```

Dadurch wird im Appletalk Netzwerk ein Drucker namens 'TricWriter' zur Verfügung gestellt. Alle Druckaufträge an diesen Drucker werden durch den Drucker-Dämon `lpd` über den Linux-Drucker `lp` (der in der Datei `/etc/printcap` definiert sein muß) ausgedruckt. Der Eintrag `op=cg` legt fest, daß der Druckauftrag unter der ID des Linux-Nutzers `cg` abgewickelt wird.

### 5.2.4 Starten der Appletalk Software.

Nun ist alles soweit konfiguriert, der erste Test kann beginnen. Zum Paket `netatalk` gehört eine Datei `rc.atalk`, die für Normalanwendungen funktionieren sollte. Alles was zu tun bleibt, ist diese Datei aufzurufen:

```
# /usr/local/atalk/etc/rc.atalk
```

und alles sollte einwandfrei laufen. Fehlermeldungen sollten keine auftreten. Der Start der Software wird, ebenso wie weitere Statusmeldungen, über die Konsole ausgegeben.

### 5.2.5 Testen der Appletalk Software.

Um zu überprüfen ob alles einwandfrei funktioniert begeben sie sich an einen ihrer Apple Rechner, öffnen sie das Apple Menue, wählen 'Chooser' aus und klicken auf AppleShare. Ihr Linux-Rechner sollte sich nun melden.

### 5.2.6 Nachteile der Appletalk Software.

- Unter Umständen müssen sie die Appletalk-Unterstützung vor der Konfiguration des IP-Netzwerkes durchführen. Gibt es beim Start des Appletalk Programmes Probleme, oder haben sie nach dessen Start Probleme mit dem IP Netzwerk, versuchen Sie die Appletalk Software *vor* der Ausführung von `/etc/rc.d/rc.inet1` zu starten.
- `afpd` (der Apple Filing Protocol Dämon) bringt die Festplatte ziemlich durcheinander. Er legt unterhalb des Mount-punktes eine Vielzahl von Verzeichnissen an: `.AppleDesktop` und `Network Trash Folder`. Weiterhin wird darin für jedes angesprochene Verzeichnis ein `.AppleDouble` angelegt, um darin Resource Forks usw. zu speichern. Überlegen sie es sich *genau*, bevor sie ihr Rootverzeichnis / exportieren. Die Aufräumarbeiten hinterher haben es in sich.
- Das Programm `afpd` erwartet von Macs Paßworte in Klartext, Sicherheitsbedenken sind also berechtigt. Benutzen sie diesen Dämon auf einer Maschine, die selber am Internet hängt, müssen Sie sich an die eigene Nase fassen, wenn hinterher jemand diese Schwachstellen ausnutzt.
- Die vorhandenen Diagnosetools wie `netstat` oder `ifconfig` unterstützen kein Appletalk. Die Information ist - unformatiert - über `/proc/net` zugänglich.

### 5.2.7 Weitere Informationsquellen.

Eine sehr viel detailliertere Beschreibung, wie man Appletalk für Linux konfiguriert, finden sie auf der Seite *Linux Netatalk HOWTO* von Anders Brownworth bei

<http://thehamptons.com/anders/netatalk/>

## 5.3 ATM

Werner Almesberger ([werner.almesberger@lrc.di.epfl.ch](mailto:werner.almesberger@lrc.di.epfl.ch)) leitet ein Projekt mit dem Ziel, auch unter Linux ATM (Asynchronous Transfer Mode ) zu unterstützen. Den aktuellen Stand des Projektes erfährt man über:

<http://lrcwww.epfl.ch/linux-atm/>

## 5.4 AX25 (AF\_AX25)

AX.25 Devicenamen sind `s10`, `s11` usw. in 2.0.\* Kernels or `ax0`, `ax1` usw.. in 2.1.\* Kernels.

**Optionen beim Kernel kompilieren:**

```
Networking options --->
  [*] Amateur Radio AX.25 Level 2
```

Die Protokolle AX25, Netrom und Rose werden von Amateurfunkern für Experimente mit Packet Radio genutzt. Eine Ausführliche Beschreibung enthält das *AX25 HOWTO*

Der Großteil der Arbeit bei der Implementation dieser Protokolle wurde von Jonathon Naylor (jsn@cs.not.ac.uk) geleistet.

## 5.5 DECNet

An der Unterstützung von DECNet wird derzeit gearbeitet. Es wird vermutlich in den späten 2.1.\* Kernels auftauchen.

## 5.6 EQL - Lastverteilung auf mehrere Leitungen.

EQL Devices haben den Namen eql. Mit Standard Kernels gibt es nur eines dieser Devices. Es nutzt mehrere Point-to-Point Verbindungen (PPP, SLIP, Plip) und faßt sie zu einer einzigen logischen Leitung zusammen, um darüber eine TCP/IP Verbindung aufzubauen. Hintergrund: Oft sind mehrere langsame Leitungen billiger als eine schnelle.

### Optionen beim Kernel kompilieren:

```
Network device support --->
  [*] Network device support
  <*> EQL (serial line load balancing) support
```

Um diesen Mechanismus zu nutzen müssen beide Maschinen EQL unterstützen. Dies ist bei Linux, neueren Dial-in Servern und Livingstone Portmasters möglich.

Um EQL richtig zu konfigurieren benötigen sie die EQL Tools:

```
sunsite.unc.edu:/pub/linux/system/Serial/eql-1.2.tar.gz
```

Die Konfiguration ist sehr logisch aufgebaut. Zunächst wird das EQL Interface konfiguriert. Es verhält sich wie jedes andere Netzwerkinterface auch, man konfiguriert IP Adresse und MTU mittels `ifconfig`, also etwa so:

```
ifconfig eql 192.168.10.1 mtu 1006
route add default eql
```

Als nächstes müssen die zu nutzenden Verbindungen von Hand aufgebaut werden. Jede denkbare Kombination von Point-to-Point Verbindungen ist möglich. Lesen sie diesbezüglich die entsprechenden Abschnitte dieses Dokumentes.

Nun müssen diese seriellen Verbindungen mit dem EQL Device verknüpfen. Man nennt das 'enslaving', der entsprechende Befehl lautet `eql_enslave`, z.B.:

```
eql_enslave eql s10 28800
eql_enslave eql ppp0 14400
```

Die angegebene ungefähre Geschwindigkeit hat keinen direkten Hardwarebezug. Der EQL Treiber nimmt diese Werte lediglich als Anhaltspunkt, um die Datagramme möglichst sinnvoll auf die vorhandenen Leitungen zu verteilen. Man kann die Werte also für das Feintuning durchaus frei verändern.

Um eine Leitung wieder aus dem EQL Verbund zu entfernen dient der Befehl `eql_emancipate`. Wieder ein Beispiel:

```
eql_emancipate eql s10
```

Das Routing wird wie für jede andere Point-to-Point Verbindung aufgesetzt. Der einzige Unterschied ist, das anstelle des seriellen Device das EQL-Device angegeben wird:

```
route add default eql0
```

Der EQL Treiber wurde von Simon Janes ([simon@ncm.com](mailto:simon@ncm.com)) entwickelt.

## 5.7 Ethernet

Die Devicenamen für Ethernet sind `eth0`, `eth1`, `eth2` usw. Der ersten gefundenen Karte wird `eth0` zugewiesen, die weiteren werden fortlaufend durchnummeriert.

Zur Inbetriebnahme einer Ethernetkarte unter Linux existiert ein eigenes HOWTO, das *Ethernet HOWTO*.

Ist der Kernel mit Unterstützung für Ethernetkarten kompiliert, ist die Konfiguration der Karte einfach. Typischerweise verwendet man etwa folgende Befehle:

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
# route add 192.168.0.0 netmask 255.255.255.0 eth0
```

Die meisten der Treiber für Ethernetkarten wurden von Donald Becker ([becker@CESDIS.gsfc.nasa.gov](mailto:becker@CESDIS.gsfc.nasa.gov)) entwickelt.

## 5.8 FDDI

Die Devicenamen für FDDI sind `fddi0`, `fddi1`, `fddi2` usw. Der ersten gefundenen Karte wird `fddi0` zugewiesen, die weiteren werden fortlaufend durchnummeriert.

Lawrence V. Stefani ([stefani@lkg.dec.com](mailto:stefani@lkg.dec.com)) hat einen Treiber für die EISA und PCI Karten der Digital Equipment Corporation entwickelt.

### Optionen beim Kernel kompilieren:

```
Network device support --->
  [*] FDDI driver support
  [*] Digital DEFEA and DEFPA adapter support
```

Ist der Kernel mit Unterstützung für FDDI kompiliert, ist die Konfiguration praktisch identisch zu derjenigen eines Ethernet Interface: Es müssen lediglich die entsprechenden FDDI-Devicenamen angegeben werden.

## 5.9 Frame Relay

Die Devicenamen für Frame Relay sind `dlci00`, `dlci01` usw. für Devices mit DLCI Encapsulation und `sdlc0`, `sdlc1` usw. für solche mit FRAD.

Frame Relay ist eine neue Netzwerktechnologie. Sie wurde speziell für Umgebungen entwickelt, in denen die Netzauslastung intermittierend ist, also oft kurzzeitig scharfe Spitzen auftreten. Für den Zugang zu einem Frame Relay Netzwerk benötigt man ein Frame Relay Access Device (FRAD). Die Frame Relay Unterstützung unter Linux hält sich an RFC-1490.

### Optionen beim Kernel kompilieren:

```
Network device support --->
  <*> Frame relay DLCI support (EXPERIMENTAL)
  (24) Max open DLCI
  (8) Max DLCI per device
  <*> SDLA (Sangoma S502/S508) support
```

Die Frame Relay Treiber und Konfigurationsprogramme wurden von Mike McLagan (mike.mclagan@linux.org) entwickelt.

Derzeit sind allerdings nur diese Karten unterstützt: Sangoma Technologies (<http://www.sangoma.com>) S502A, S502E und S508.

Um die FRAD und DLCI Devices zu konfigurieren benötigen Sie spezielle Programme, die *Frame Relay Configuration Tools*. Diese bekommen Sie bei

```
ftp.invlogic.com:/pub/linux/fr/frad-0.15.tgz
```

Kompilierung und Installation der Tools ist eigentlich kein Problem, allerdings gibt es kein zentrales Makefile. Dadurch ist einige Handarbeit notwendig:

```
# cd /usr/src
# tar xvfz ../frad-0.15.tgz
# cd frad-0.15
# for i in common dlci frad; do cd $i; make clean; make; cd ..; done
# mkdir /etc/frad
# install -m 644 -o root -g root bin/*.sfm /etc/frad
# install -m 700 -o root -g root frad/fradcfg /sbin
# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Nach der Installation müssen sie die Datei `/etc/frad/router.conf` anlegen. Dafür ist folgende Vorlage hilfreich (es ist eine abgeänderte Version der Beispieldatei des Paketes):

```
# /etc/frad/router.conf
# Dies ist eine Beispielkonfiguration fuer Frame Relay.
# Alle moeglichen Eintraege sind aufgefuehrt, die Standardeinstellungen
# basieren auf dem Code des DOS-Treibers fuer die Karte
# S502A von Sangoma.
#
# Ein '#' irgendwo in der Zeile leitet einen Kommentar ein
# Leerzeilen werden ignoriert (TAB ist auch erlaubt).
# Unbekannte Eintraege [] oder Zeichen werden ignoriert.
#

[Devices]
Count=1                # Anzahl zu konfigurierender Devices
Dev_1=sdl1             # Name eines Device
#Dev_2=sdl1            # Name eines Device

# An dieser Stelle angegeben, gelten die Eintraege fuer alle Devices.
# Sie koennen fuer einzelne Karten in den entsprechenden Abschnitten
# veraendert werden.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Maximum transmit IFrame length, default is 4096
# T391=10               # T391 value      5 - 30, default is 10
# T392=15               # T392 value      5 - 30, default is 15
# N391=6                # N391 value      1 - 255, default is 6
# N392=3                # N392 value      1 - 10, default is 3
# N393=4                # N393 value      1 - 10, default is 4
```

```
# An dieser Stelle angegeben, werden Standardwerte fuer alle Devices
# festgelegt.
# CIRfwd=16          # CIR forward   1 - 64
# Bc_fwd=16         # Bc forward   1 - 512
# Be_fwd=0          # Be forward   0 - 511
# CIRbak=16         # CIR backward 1 - 64
# Bc_bak=16         # Bc backward  1 - 512
# Be_bak=0          # Be backward  0 - 511

#
#
# Device spezifische Konfiguration
#
#
#
# Das erste Device ist eine Sangoma S502E
#
[sdla0]
Type=Sangoma          # Art des Device
                     # SANGOMA ist bekannt

#
# Diese Eintraege sind spezifisch fuer Sangoma
#
# Typ der Sangoma Karte - S502A, S502E, S508
Board=S502E
#
# Name der Test-Firmware fuer das Sangoma Board
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# Name der FR Firmware
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360              # Port fuer diese Karte
Mem=C8               # Address fuer Memory Window, A0-EE
IRQ=5                # IRQ Nummer, fuer S502A nicht angeben
DLCIs=1              # Anzahl der DLCI's an diesem Device
DLCI_1=16            # DLCI #1's number, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Hier angegeben, gelten die Eintraege nur fuer die jeweilige Karte und
# ueberschreiben im globalen Teil gemachte Einstellungen.
#
# Access=CPE         # CPE oder NODE, Default ist CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal     # External oder Internal, Default ist Internal
# Baud=128           # Angegebene Baud Rate des angeschlossenen CSU/DSU
# MTU=2048           # Maximale IFrame Laenge, Default ist 4096
# T391=10            # T391 value     5 - 30, Default ist 10
# T392=15            # T392 value     5 - 30, Default ist 15
# N391=6             # N391 value     1 - 255, Default ist 6
# N392=3             # N392 value     1 - 10, Default ist 3
```

```

# N393=4                # N393 value    1 - 10, Default ist 4

#
# Die zweite Karte ist irgend eine andere Karte
#
# [sdlal]
# Type=FancyCard        # Art des Device
# Board=                 # Typ des Sangoma board
# Key=Value              # Eintraege spezifisch fuer dieses Device

#
# DLCI Default Konfigurationsparameter
# Diese koennen in den jeweiligen spezifischen Abschnitten
# ueberschrieben werden.
#
CIRfwd=64                # CIR forward    1 - 64
# Bc_fwd=16              # Bc forward     1 - 512
# Be_fwd=0                # Be forward     0 - 511
# CIRbak=16              # CIR backward   1 - 64
# Bc_bak=16              # Bc backward    1 - 512
# Be_bak=0                # Be backward    0 - 511

#
# DLCI Konfiguration
# Alle Eintraege sind optional.  Namenkonvention ist:
# [DLCI_D<devicenum>_<DLCI_Num>]
#
[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Von Sangoma definierte Flags sind: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Von Sangoma definierte Flags sind: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0

```

Ist die Datei `/etc/frad/router.conf` angelegt, bleibt nur noch die Konfiguration der eigentlichen Devices. Dies ist nicht viel schwieriger als die übliche Konfiguration eines Netzwerk Devices. Man muß nur daran denken, die

FRAD Devices vor den DLCI Devices zu konfigurieren.

```
# Konfiguriere FRAD Hardware und DLCI Parameter
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Aktiviere FRAD Device
ifconfig sdla0 up
#
# Konfiguriere das DLCI Encapsulation Interface und Routing
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add 192.168.10.0 netmask 255.255.255.0 dlci00
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

## 5.10 IP Accounting

IP Accounting im Kernel erlaubt es, Daten über die Nutzung des Netzwerkes zu sammeln und zu analysieren. Die Daten umfassen die Anzahl der Pakete bzw. Bytes seit dem letzten Reset der Zähler. Es können eine Vielzahl von Regeln festgelegt werden, um die verschiedenen Zähler den eigenen Bedürfnissen anzupassen.

### Optionen beim Kernel kompilieren:

```
Networking options --->
[*] IP: accounting
```

Nach Kompilierung und Installation des Kernels benötigen sie das Programm `ipfwadm`, um das IP Accounting zu konfigurieren. Es gibt eine Menge unterschiedlicher Wege, die Accounting Information in verschiedene Bereiche aufzuspalten. Hier ist ein einfaches Beispiel als Anregung, für weitergehende Informationen sollten Sie die Hilfeseite zu `ipfwadm` lesen.

Das Szenario für das Beispiel ist folgendes: Ein lokales Ethernet ist über eine serielle PPP-Leitung mit dem Internet verbunden. Im Internet steht ein Rechner, der einige Dienste zur Verfügung stellt. Sie sind daran interessiert zu erfahren, welchen Anteil der Auslastung durch die Dienste `telnet`, `rlogin`, `ftp` und `WWW` verursacht wird.

Eine entsprechende Konfiguration sieht so aus:

```
#
# Loeschen der bestehenden Accounting Regeln
ipfwadm -A -f
#
# Neue Regeln fuer das lokale Ethernet Segment
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
ipfwadm -A in -a -P tcp -D 44.136.8.96/29
ipfwadm -A out -a -P tcp -D 44.136.8.96/29
ipfwadm -A in -a -P udp -D 44.136.8.96/29
```



```

ipfwadm -A out -a -P udp -D 44.136.8.96/29
ipfwadm -A in -a -P icmp -D 44.136.8.96/29
ipfwadm -A out -a -P icmp -D 44.136.8.96/29
#
# Default Regeln
ipfwadm -A in -a -P tcp -D 0/0 20
ipfwadm -A out -a -P tcp -S 0/0 20
ipfwadm -A in -a -P tcp -D 0/0 23
ipfwadm -A out -a -P tcp -S 0/0 23
ipfwadm -A in -a -P tcp -D 0/0 80
ipfwadm -A out -a -P tcp -S 0/0 80
ipfwadm -A in -a -P tcp -D 0/0 513
ipfwadm -A out -a -P tcp -S 0/0 513
ipfwadm -A in -a -P tcp -D 0/0
ipfwadm -A out -a -P tcp -D 0/0
ipfwadm -A in -a -P udp -D 0/0
ipfwadm -A out -a -P udp -D 0/0
ipfwadm -A in -a -P icmp -D 0/0
ipfwadm -A out -a -P icmp -D 0/0
#
# Auflisten der Regeln
ipfwadm -A -l -n
#

```

Der letzte Befehl zeigt eine Auflistung aller Accounting Regeln und zeigt die aufsummierten Zahlenwerte an.

Ein wichtiger Punkt bei der Auswertung der Accounting Informationen ist, daß **die Zähler für alle zutreffenden Regeln erhöht werden**. Für eine genaue, differentielle Analyse muß man also ein wenig Rechnen. Um z.B. herauszufinden, welcher Datenanteil nicht von ftp, telnet, rlogin oder WWW herrührt, müssen die Summe der Zahlenwerte der einzelnen Ports subtrahiert werden von der Regel, die alle Ports umfaßt:

```

# ipfwadm -A -l -n
IP accounting rules
pkts bytes dir prot source destination ports
  0    0 in  tcp  0.0.0.0/0 44.136.8.96/29 * -> 20
  0    0 out tcp  44.136.8.96/29 0.0.0.0/0 20 -> *
  0    0 in  tcp  0.0.0.0/0 44.136.8.96/29 * -> 23
  0    0 out tcp  44.136.8.96/29 0.0.0.0/0 23 -> *
 10  1166 in  tcp  0.0.0.0/0 44.136.8.96/29 * -> 80
 10   572 out tcp  44.136.8.96/29 0.0.0.0/0 80 -> *
242  9777 in  tcp  0.0.0.0/0 44.136.8.96/29 * -> 513
220 18198 out tcp  44.136.8.96/29 0.0.0.0/0 513 -> *
252 10943 in  tcp  0.0.0.0/0 44.136.8.96/29 * -> *
231 18831 out tcp  0.0.0.0/0 44.136.8.96/29 * -> *
  0    0 in  udp  0.0.0.0/0 44.136.8.96/29 * -> *
  0    0 out udp  0.0.0.0/0 44.136.8.96/29 * -> *
  0    0 in  icmp 0.0.0.0/0 44.136.8.96/29 *
  0    0 out icmp 0.0.0.0/0 44.136.8.96/29 *
  0    0 in  tcp  0.0.0.0/0 0.0.0.0/0 * -> 20
  0    0 out tcp  0.0.0.0/0 0.0.0.0/0 20 -> *
  0    0 in  tcp  0.0.0.0/0 0.0.0.0/0 * -> 23
  0    0 out tcp  0.0.0.0/0 0.0.0.0/0 23 -> *
 10  1166 in  tcp  0.0.0.0/0 0.0.0.0/0 * -> 80
 10   572 out tcp  0.0.0.0/0 0.0.0.0/0 80 -> *
243  9817 in  tcp  0.0.0.0/0 0.0.0.0/0 * -> 513
221 18259 out tcp  0.0.0.0/0 0.0.0.0/0 513 -> *
253 10983 in  tcp  0.0.0.0/0 0.0.0.0/0 * -> *

```

```

231 18831 out tcp 0.0.0.0/0          0.0.0.0/0          * -> *
    0      0 in  udp 0.0.0.0/0          0.0.0.0/0          * -> *
    0      0 out udp 0.0.0.0/0          0.0.0.0/0          * -> *
    0      0 in  icmp 0.0.0.0/0        0.0.0.0/0          *
    0      0 out icmp 0.0.0.0/0      0.0.0.0/0          *
#

```

## 5.11 IP Aliasing

Es gibt einige Anwendungen bei denen es hilfreich ist, wenn man einem einzelnen Netzwerk-Device mehrere IP Adressen zuweisen kann. Provider für Internet Dienste verwenden dies häufig, um ihren Kunden speziell angepaßte WWW- und FTP-Dienste anzubieten.

### Optionen beim Kernel kompilieren:

```

Networking options --->
....
[*] Network aliasing
....
<*> IP: aliasing support

```

Die Konfiguration für IP Aliasing ist sehr einfach. Die Aliases werden virtuellen Netzwerk Devices zugewiesen, die an das tatsächliche Device gekoppelt sind. Eine einfache Namenskonvention für diese Devices ist <Devicename>:<virtuelle Dev Nummer>, also z.B. eth0:0, ppp0:10 usw.

Als Beispiel nehmen wir ein Ethernet Netzwerk mit zwei IP Subnetzwerken an. Um beide gleichzeitig über eine Netzwerkkarte anzusprechen dienen folgende Befehle:

```

#
# ifconfig eth0:0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0:0
#
# ifconfig eth0:1 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
#

```

Um einen Alias zu löschen, hängen sie einfach ein '-' an das Ende seines Namens an:

```

# ifconfig eth0:0- 0

```

Alle mit diesem Device verbundenen Routes werden automatisch ebenfalls gelöscht.

## 5.12 IP Firewall

Alles was mit IP Firewall und Firewalls allgemein zu tun hat wird ausführlich im *Firewall HOWTO* erläutert. Ein IP Firewall erlaubt es, den Rechner oder ein ganzes Netzwerk gegen unerlaubte Netzzugriffe abzuschotten, indem Datenpakete von und zu angegebenen IP-Adressen gefiltert werden. Es existieren drei unterschiedliche Klassen für Regeln: Incoming Filter, Outgoing Filter und Forward Filter. *Incoming* Filter werden auf Datenpakete angewandt, die über eine Netzwerkschnittstelle empfangen werden. *Outgoing* Filter gelten für Datenpakete, die über eine Netzwerkschnittstelle ausgegeben werden. *Forward* Filter werden auf Datenpakete angewandt, die zwar angenommen werden, aber nicht für den eigenen Rechner bestimmt sind, also solche, die gerouted werden.

### Optionen beim Kernel kompilieren:

```

Networking options --->
  [*] Network firewalls
  ....
  [*] IP: forwarding/gatewaying
  ....
  [*] IP: firewalling
  [ ] IP: firewall packet logging

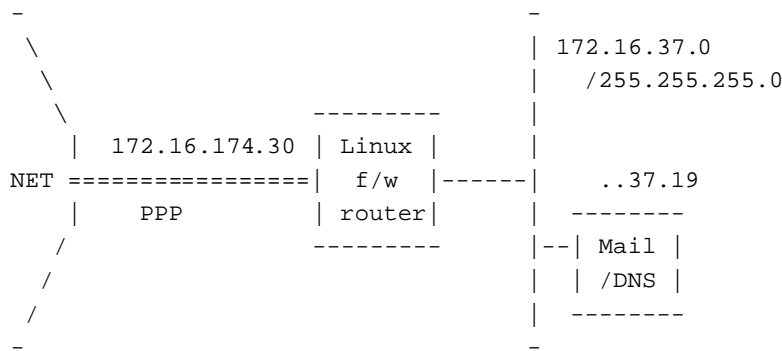
```

Die Konfiguration eines IP Firewall wird mit dem Befehl `ipfwadm` durchgeführt. Wie bereits erwähnt bin ich kein Experte in Sachen Sicherheit. Obwohl hier ein Beispiel für die Konfiguration angegeben wird, sollten Sie weitere Nachforschungen auf diesem Gebiet anstellen und ihre eigenen Regeln zusammensuchen, wenn Sie wirklich auf Sicherheit bedacht sind.

Am weitesten Verbreitet ist die Benutzung von IP Firewall, um einen Linux-Rechner als Router und Firewall Gateway für ein lokales Netzwerk einzusetzen und dieses gegen unerlaubten Zugriff von außerhalb zu sichern.

Die folgende Konfiguration basiert auf einem Beitrag von Arnt Gulbrandsen (`agulbra@troll.no`).

Das Beispiel beschreibt die Konfiguration der Firewall-Regeln des Linux Firewall/Router Rechners aus folgendem Schaubild:



Die folgenden Befehle gehören eigentlich in eine `rc`-Datei, so daß sie automatisch bei jedem Systemstart ausgeführt werden. Um maximale Sicherheit zu erreichen, sollten sie *nach* der Konfiguration der Netzwerk Devices, aber *vor* deren Aktivierung ausgeführt werden. Dadurch wird ein Einbruch während des Bootens unterbunden.

```

#!/bin/sh

# Loeschen der Forwarding Regeln
# Default Policy auf 'accept'
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# .. ebenso fuer 'Incoming'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# Als erstes das PPP Interface schlie{\ss}en
# besser waere hier '-a deny' anstelle von '-a reject -y', aber dann
# waere es auch nicht mehr moeglich, ueber dieses Interface selber eine
# Verbindung aufzubauen.
# Das -o veranlasst, dass alle geblockten Datagramme protokolliert
# werden. Das verbraucht viel Plattenplatz, andernfalls ist man aber
# ueber Angriffsversuche oder Fehlkonfiguration im Unklaren.
#

```

```
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Einige offensichtlich falsche Pakete werden sofort abgewiesen:
# Von multicast/anycast/broadcast Adressen sollte nicht kommen.
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# und auch vom Loopback Netzwerk sollten keine Pakete auf der Leitung
# erscheinen.
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# Eingehende SMTP und DNS Verbindungen werden akzeptiert, aber nur an
# den Mail/Nameserver
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS verwendet UDP und TCP, deshalb muss das auch freigegeben werden
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# allerdings keine 'Antworten' von gefaehrlichen Ports wie NFS und
# Larry McVoy's NFS extension. Wer SQUID verwendet, sollte dessen Ports
# hier ebenfalls angeben
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
-D 172.16.37.0/24 2049 2050

# Antworten an andere User Ports sind OK
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
-D 172.16.37.0/24 53 1024:65535

# Eingehende Verbindungen mit identd werden geblockt.
# Hier wird 'reject' verwendet, damit dem anderen Rechner sofort
# mitgeteilt wird, das weitere Versuche sinnlos sind. Andernfalls
# wuerden Verzoegerungen durch timeouts von ident auftreten.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Einige Standard-Dienste werden fuer Verbindungen von den Netzwerken
# 192.168.64 und 192.168.65 akzeptiert; das sind Freunde, denen wir trauen.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
-D 172.16.37.0/24 20:23

# Alles von innerhalb des lokalen Netzes wird akzeptiert und
# weitergeleitet.
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# Alle anderen eingehenden TCP Verbindungen werden verweigert und
# protokolliert. (Falls FTP nicht funktioniert, haengen Sie ein
# 1:1023 an).
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24
```

```
# ... ebenfalls fuer UDP
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Gute Firewall Konfigurationen sind etwas trickreich. Dieses Beispiel sollte einen brauchbaren Anfang liefern. Die Hilfeseite zu `ipfwadm` gibt weitere Unterstützung bei der Konfiguration. Wenn Sie vorhaben, einen Firewall einzurichten, erkundigen Sie sich auch bei vertrauenswürdigen Bekannten und sammeln sie soviel Hinweise und Ratschläge wie möglich. Suchen sie auch jemanden, der ein paar Zuverlässigkeits- und Funktionstests von außerhalb durchführt.

### 5.13 IPX (AF\_IPX)

Das IPX Protokoll wird hauptsächlich in lokalen Netzwerken unter Novell Netware(tm) verwendet. Linux unterstützt dieses Protokoll und kann als Endpunkt oder Router für IPX verwendet werden.

**Optionen beim Kernel kompilieren:**

```
Networking options --->
  [*] The IPX protocol
  [ ] Full internal IPX network
```

IPX Protokoll und NCPFS werden ausführlich im *IPX HOWTO* behandelt.

### 5.14 IPv6

Da hat man nun gerade geglaubt, IP Netzwerke ansatzweise zu verstehen, und nun werden die Regeln geändert! IPv6 ist eine abgekürzte Form für die Version 6 des Internet Protokolls. IPv6 wurde vorrangig entwickelt, um den Befürchtungen der Internet Gemeinde entgegenzuwirken, daß es bald einen Engpaß bei den IP Adressen gäbe. IPv6 Adressen sind 32 Byte, also 128 Bit, lang. Außerdem enthält IPv6 einige weitere Änderungen, vorrangig Vereinfachungen, die ein IPv6 Netzwerk einfacher verwaltbar machen als ein IPv4 Netzwerk.

Die Kernels der Version 2.1.\* enthalten bereits eine funktionierende, wenn auch noch unvollständige Implementation von IPv6.

Wenn Sie mit dieser neuen Generation der Internet Technologie experimentieren wollen, sollten Sie das IPv6-FAQ lesen. Es ist von

<http://www.terra.net/ipv6>

erhältlich.

### 5.15 ISDN

Das Integrated Services Digital Network (ISDN) umfaßt eine Reihe von Standards, die ein Vielzweck Digitales Netzwerk definieren. Ein ISDN 'Anruf' baut eine synchrone Point-to-Point Verbindung, einen *Service*, zur angerufenen Seite auf. ISDN wird für gewöhnlich auf Hochgeschwindigkeitsleitungen verwendet, die in mehrere diskrete Kanäle aufgeteilt ist. Es gibt zwei unterschiedliche Typen dieser Kanäle, 'B-Kanäle', die die eigentlichen Benutzerdaten übertragen, und einen einzelnen 'C-Kanal', über den ISDN Kontrollinformationen zum Verbindungsaufbau und andere Funktionen überträgt. In Australien wird ISDN z.B. über eine 2Mbps Leitung übertragen, die in 30 B-Kanäle zu 64kbps sowie einen D-Kanal mit ebenfalls 64kbps aufgespalten ist. Zu jedem Zeitpunkt können beliebig viele dieser Kanäle in jeder Kombination benutzt werden. Also z.B. gleichzeitig 30 Verbindungen (64kbps) zu 30 verschiedenen Zielen aufbauen, oder 15 Verbindungen (128kbps) zu 15 unterschiedlichen Zielen, oder eben auch nur eine kleine Zahl von Verbindungen, wobei ein Teil der Kanäle unbenutzt bleibt. Ein Kanal kann für eingehende oder ausgehende Verbindungen genutzt werden. Das ursprüngliche Ziel hinter ISDN war es, den Telekommunikationsgesellschaften die

Möglichkeit zu geben, über eine einzelne Leitung sowohl Telefondienste (als digitalisierte Sprache) als auch Datendienste anzubieten, ohne daß Änderungen in der Konfiguration notwendig werden.

Es gibt unterschiedliche Wege, einen Rechner an ISDN anzuschließen. Eine Möglichkeit stellt ein 'Terminal Adapter' dar. Dieser wird direkt an die Netzwerk-Endstelle der Telekom angeschlossen und stellt eine Anzahl von seriellen Schnittstellen zur Verfügung. Eine dieser Schnittstellen wird dazu benutzt, um Kommandos zum Verbindungsaufbau, Konfiguration usw. zu übermitteln, die übrigen sind mit den Netzwerk-Devices gekoppelt, über die die Datenverbindungen aufgebaut werden. In dieser Umgebung funktioniert Linux ohne Veränderungen, die Ports am Terminal Adapter werden einfach wie eine normale serielle Schnittstelle behandelt. Eine andere Möglichkeit besteht im Einsatz einer speziellen Einsteckkarte für den Rechner. Dafür sind die ISDN Treiber im Kernel vorgesehen. In diesem Fall werden Protokolle und Verbindungsaufbau von der Software des Rechners überwacht.

#### Optionen beim Kernel kompilieren:

```
ISDN subsystem --->
  <*> ISDN support
  [ ] Support synchronous PPP
  [ ] Support audio via ISDN
  < > ICN 2B and 4B support
  < > PCBIT-D support
  < > Teles/NICCY1016PC/Creatix support
```

Linux unterstützt eine Reihe unterschiedlicher ISDN Karten:

- ICN 2B und 4B
- Octal PCBIT-D
- Teles ISDN-Karten und Kompatible

Für einige dieser Karten ist zusätzliche Software nötig, um sie zu betreiben. Diese muß mit speziellen Programmen geladen werden.

Weitere Details zur Konfiguration von ISDN unter Linux befinden sich im Verzeichnis `/usr/src/linux/Documentation/isdn`, außerdem existiert das FAQ *isdn4linux*, zu beziehen bei

<http://www.lrz-muenchen.de/~ui161ab/www/isdn/>

Es gibt dort eine deutsche und eine englische Version.

**Ein Hinweis zu PPP.** PPP ist generell für den Betrieb sowohl über synchrone wie auch über asynchrone serielle Verbindungen ausgelegt. Der normalerweise in Linux-Distributionen enthaltene Dämon `pppd` unterstützt jedoch nur den asynchronen Modus. Wenn sie PPP über ihre ISDN Verbindung benutzen wollen, benötigen sie eine speziell angepaßte Version. Nähere Details dazu finden sie in den oben erwähnten Quellen.

## 5.16 IP Masquerade

Viele Personen setzen eine einfache Einwahlverbindung als Zugang zum Internet ein. Hierbei wird dem einwählenden Rechner praktisch immer genau eine einzige IP Adresse zugewiesen. Das ist normalerweise ausreichend, um einem einzelnen Rechner vollen Zugang zu den Möglichkeiten des Internet zu geben. IP Masquerade erlaubt es nun auch, daß beliebig viele Rechner (z.B. ein lokales Netzwerk) diese eine IP Adresse verwenden, indem man erreicht, daß sie nach außen so aussehen wie der eine Rechner, dessen IP Adresse verwandt wird - deshalb der Name 'Maskerade'. Ein kleiner Nachteil dabei ist allerdings, daß dieses Masquerading immer nur in eine Richtung funktioniert. D.h. der maskierte Rechner kann zwar Verbindungen nach außen aufbauen, er kann aber keine Anfragen/Verbindungen von außenliegenden Rechnern empfangen. Deshalb funktionieren einige Dienste wie `talk` nicht, andere wie z.B. `ftp` müssen speziell auf passiven Modus (PASV) konfiguriert werden, damit sie funktionieren. Zum Glück sind aber Standard-Dienste wie `telnet`, `irc` und WWW davon nicht betroffen.

**Optionen beim Kernel kompilieren:**

```
Code maturity level options --->
  [*] Prompt for development and/or incomplete code/drivers
Networking options --->
  [*] Network firewalls
  ....
  [*] TCP/IP networking
  [*] IP: forwarding/gatewaying
  ....
  [*] IP: masquerading (EXPERIMENTAL)
```

Auf dem Linux Rechner wird SLIP oder PPP ganz normal konfiguriert, wie für einen Einzelrechner. Außerdem besitzt der Rechner aber eine weitere Netzwerkschnittstelle, z.B. eine Ethernetkarte, über die er mit dem lokalen Netzwerk verbunden ist. An diesem Netz hängen auch die Rechner, die 'maskiert' werden sollen. Jeder dieser anderen Rechner muß nun zunächst die IP Adresse des Linux Rechners als Gateway bzw. Router konfigurieren.

Eine typische Konfiguration sieht etwa so aus:

```

-
  \
   \
    |
NET =====| Linux | .1.1 |
    |         | masq  |-----|
    |         | router|         |
    |         |-----|         |
  /         | host  |         |
 /         |-----|         |
-

```

Die wichtigsten Konfigurationsbefehlein diesem Fall sind:

```
# Netzwerk Route fuer Ethernet
route add 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default Route fuer den Rest des Internet
route add default ppp0
#
# Alle Hosts auf dem Netzwerk 192.168.1/24 werden maskiert
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Weitere Informationen über IP Masquerade unter Linux enthält die IP Masquerade Resource Page:

<http://www.hwy401.com/achau/ipmasq/>

**5.17 IP Transparent Proxy**

IP Transparent Proxy ermöglicht es, Anfragen für Server oder Dienste auf anderen Rechnern auf die lokale Maschine umzulenken. Dies ist z.B. sinnvoll, wenn ein Linux Rechner als Router und Proxy Server eingesetzt wird. In diesem Fall werden alle Anfragen nach nichtlokalen Diensten an den lokalen Proxy weitergeleitet.

**Optionen beim Kernel kompilieren:**

```
Code maturity level options --->
  [*] Prompt for development and/or incomplete code/drivers
```

```
Networking options --->
  [*] Network firewalls
  ....
  [*] TCP/IP networking
  ....
  [*] IP: firewalling
  ....
  [*] IP: transparent proxy support (EXPERIMENTAL)
```

Die Konfiguration von IP Transparent Proxy wird mit Hilfe des Befehles `ipfwadm` durchgeführt, zum Beispiel so:

```
ipfwadm -I -a accept -D 0/0 80 -r 8080
```

Dadurch wird jede Verbindungsversuch mit dem Port 80 (www) eines beliebigen Rechners auf den Port 8080 des lokalen Rechners umgeleitet. Dadurch kann man z.B. sicherstellen daß jeglicher WWW Verkehr auf dem Netzwerk automatisch über ein lokales Cache Programm geleitet wird.

## 5.18 Mobile IP

Der Ausdruck "IP mobility" beschreibt die Fähigkeit eines Rechners, seine Verbindung zum Internet an unterschiedliche Punkte zu verlagern, ohne dabei seine IP Adresse zu ändern oder die Verbindung zu verlieren. Normalerweise ändert sich die IP Adresse eines Rechners, wenn er an einer anderen Stelle (z.B. in über ein anderen Netzwerk) an das Internet angekoppelt wird. Mobile IP umgeht dieses Problem, indem dem Rechner eine feste IP Adresse zugeordnet wird und jeglicher Datenverkehr zu diesem Rechner durch IP Encapsulation (Tunneling) an die momentan tatsächlich genutzte IP Adresse umgeleitet wird.

In einem derzeit in Entwicklung befindlichen Projekt sollen alle notwendigen Programme für IP Mobility unter Linux zusammengetragen werden. Den gegenwärtigen Stand der Dinge erfahren Sie auf der *Linux Mobile IP Home Page*:

```
http://anchor.cs.binghamton.edu/~mobileip/
```

## 5.19 Multicast

Mit IP Multicast ist es möglich, Datenpakete gleichzeitig an beliebig viele Rechner auf verschiedenen IP Netzwerken zu routen. Dieser Mechanismus wird ausgenutzt, um Internet-weite Verteilung von z.B. Audio- oder Videodaten zu ermöglichen.

### Optionen beim Kernel kompilieren:

```
Networking options --->
  [*] TCP/IP networking
  ....
  [*] IP: multicasting
```

Ein paar spezielle Programme sowie einige kleinere Konfigurationsänderungen des Netzwerkes sind nötig, um diese Möglichkeiten auszunutzen. Weitere Informationen zu Installation und Konfiguration findet man z.B. bei

```
http://www.teksouth.com/linux/multicast/
```

## 5.20 NetRom (AF\_NETROM)

Die NetRom Devices sind `nr0`, `nr1` usw.

### Optionen beim Kernel kompilieren:



```
Networking options --->
[*] Amateur Radio AX.25 Level 2
[*] Amateur Radio NET/ROM
```

Die Protokolle AX25, Netrom und Rose werden von Amateurfunkern für Experimente mit Packet Radio genutzt. Eine Ausführliche Beschreibung enthält das *AX25 HOWTO*.

Der Großteil der Arbeit bei der Implementation dieser Protokolle wurde von Jonathon Naylor ([jsn@cs.not.ac.uk](mailto:jsn@cs.not.ac.uk)) geleistet.

## 5.21 PLIP

Die Namen der PLIP Devices sind `plip0`, `plip1` usw. Das erste Device erhält die Nummer 0, die weiteren werden fortlaufend durchnummeriert.

### Optionen beim Kernel kompilieren:

```
Networking options --->
<*> PLIP (parallel port) support
```

*PLIP* (Parallel Line IP) wird – wie *SLIP* – benutzt, um eine Point-to-Point Netzwerkverbindung zwischen zwei Rechnern herzustellen. Im Unterschied zu *SLIP* werden dazu jedoch die Parallelports der Rechner verwendet. Da dabei mehr als ein Bit gleichzeitig übertragen werden kann, lassen sich mit *PLIP* höhere Datenübertragungsraten erreichen. Außerdem lassen sich selbst die einfachsten parallelen Anschlüsse, die Druckerports, verwenden.

Aber Vorsicht: Manche Laptops verwenden Chipsätze, mit denen *PLIP* nicht verwendet werden kann: Sie lassen bestimmte Kombinationen von Signalen, die *PLIP* zum Funktionieren benötigt, nicht zu, da sie von Druckern nicht verwendet werden.

Die *PLIP* Schnittstelle von Linux ist kompatibel zum *Crynowyr Packet Driver PLIP*, d.h. man kann damit eine vollwertige TCP/IP Verbindung zwischen seinem Linux Rechner und einem DOS-Rechner aufbauen.

Beim Kompilieren des Kernels sollte man einen Blick in die Datei `/usr/src/linux/driver/net/CONFIG` werfen. Sie enthält Definitionen für den *PLIP* Timer in mS. Die Standardwerte sind zwar meist OK, insbesondere bei langsamen Rechnern wird man sie aber unter Umständen erhöhen müssen – auf dem *schnellen* Rechner.

Der Treiber geht von folgenden Einstellungen aus:

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

Entspricht ihr Parallelports keiner dieser Möglichkeiten, können die Werte mit dem Befehl `ifconfig` und der Option `irq` geändert werden. Achten Sie auch darauf, daß die IRQ's für den Parallelport im BIOS aktiviert sind.

Zu Konfiguration des *PLIP* Interface müssen die folgenden Befehle in der für das Netzwerk zuständigen `rc`-Datei hinzugefügt werden:

```
#
# Attach a PLIP interface
#
# Konfiguriere den ersten Parallelport als PLIP Device
/sbin/ifconfig plip0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# Ende PLIP
```

Dabei ist

#### **IPA.IPA.IPA.IPA**

ihre IP Adresse.

#### **IPR.IPR.IPR.IPR**

die IP Adresse des anderen Rechners.

Der Parameter *pointpoint* hat dieselbe Bedeutung wie bei SLIP: Es wird die Adresse des Rechners am anderen Ende der Verbindung angegeben.

Ansonsten kann man ein PLIP Interface genau wie ein SLIP Interface behandelt, einzig *dip* oder *slattach* brauchen und können nicht verwendet werden.

### **5.21.1 Ein Kabel für PLIP.**

PLIP wurde so ausgelegt, daß es dieselben Verbindungskabel verwendet wie die normalerweise auf DOS-Rechnern verwendeten Datenübertragungsprogramme.

Die Pinbelegung (aus `/usr/src/linux/drivers/net/plip.c`) sieht folgendermassen aus:

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*

Hinweis: Die mit einem Stern \* gekennzeichneten Pins dürfen nicht verbunden werden. Zusätzliche Erdungsanschlüsse sind 18,19,20,21,22,23 und 24.

Geschirmte Kabel sollten **nur auf einer Seite** mit dem Metall des Steckers verbunden werden.

**Vorsicht: Ein falsch verdrahtetes PLIP-Kabel kann ihre Controller Karte zerstören!** Seien Sie sehr vorsichtig, und überprüfen sie jede Verbindung doppelt um unnötigen Ärger zu vermeiden.

Obwohl man PLIP Verbindungen teilweise auch über lange Distanzen verwenden kann sollten Sie das nach Möglichkeit vermeiden. Die Spezifikationen erlauben eine Kabellänge von etwa einem Meter. Wenn Sie dennoch längere Kabel verwenden wollen, achten Sie besonders auf elektromagnetische Störeinstreuungen (Blitz, andere Stromkabel, Radiosender), da auch dadurch eine Beeinträchtigung der Verbindung bis hin zur Beschädigung des Controllers möglich ist. Wenn sie wirklich eine Verbindung über größere Distanzen herstellen wollen oder müssen, kaufen Sie lieber zwei billige Ethernet-Karten und ein Koaxial-Kabel

## 5.22 PPP

Die Namen der PPP Devices sind `ppp0`, `ppp1` usw. Die Devices werden fortlaufend durchnummeriert, beginnend mit 0 für das erste konfigurierte Device.

### Optionen beim Kernel kompilieren:

```
Networking options --->
  <*> PPP (point-to-point) support
```

Die Konfiguration von PPP wird im *PPP HOWTO* beschrieben.

### 5.22.1 Permanente Netzverbindungen mit `pppd`.

Falls Sie sich in der glücklichen Lage befinden eine mehr oder weniger dauerhafte Netzanbindung zu haben gibt es eine sehr einfache Möglichkeit, daß der Rechner automatisch eine neue PPP Verbindung aufbaut, wenn diese zusammenbrechen sollte.

Dabei muß PPP derart konfiguriert werden, daß es vom Superuser `root` durch einen einfachen Befehl gestartet werden kann:

```
# pppd
```

**Stellen Sie sicher** daß sie in der Datei `/etc/ppp/options` die Option `-detach` konfiguriert haben. Dann fügen sie die folgende Zeile bei den `getty`-Definitionen in die Datei `/etc/inittab` ein:

```
pd:23:respawn:/usr/sbin/pppd
```

Dadurch wird der Dämon `pppd` laufend von `init` überwacht und im Falle eines Verbindungsabbruches automatisch neu gestartet.

## 5.23 Rose protocol (`AF_ROSE`)

Die Namen der Rose Devices sind `rs0`, `rs1` usw. Rose wird nur in den Entwickler-Kernels `2.1.*` unterstützt.

### Optionen beim Kernel kompilieren:

```
Networking options --->
  [*] Amateur Radio AX.25 Level 2
  <*> Amateur Radio X.25 PLP (Rose)
```

Die Protokolle AX25, Netrom und Rose werden von Amateurfunkern für Experimente mit Packet Radio genutzt. Eine Ausführliche Beschreibung enthält das *AX25 HOWTO*.

Der Großteil der Arbeit bei der Implementation dieser Protokolle wurde von Jonathon Naylor (`jsn@cs.not.ac.uk`) geleistet.

## 5.24 SAMBA - 'NetBEUI', 'NetBios' Unterstützung.

SAMBA ist eine Implementation des Session Management Block Protokolles. Mit SAMBA ist es möglich, daß Microsoft- und andere Systeme die Platten des Linux-Rechners mounten können und dessen Drucker verwenden.

SAMBA und seine Konfiguration werden ausführlich im *Linux Samba HOWTO* beschrieben.

## 5.25 SLIP Klient

Die Namen der SLIP Devices sind `s10`, `s11` usw. Das erste konfigurierte Device erhält die Nummer 0, weitere werden fortlaufend durchnummeriert.

### Optionen beim Kernel kompilieren:

```
Network device support --->
  [*] Network device support
  <*> SLIP (serial line) support
  [ ] CSLIP compressed headers
  [ ] Keepalive and linefill
  [ ] Six bit SLIP encapsulation
```

SLIP (Serial Line IP) ermöglicht TCP/IP Verbindungen über serielle Leitungen wie Telefonleitungen (mit Modem) oder geleaste Leitungen. Um es zu benutzen benötigt man einen *SLIP-Server* möglichst in der näheren Umgebung. Viele Universitäten und einige Firmen bieten einen solchen Service an.

SLIP verwendet die serielle Schnittstelle des Rechners, um Datenpakete zu versenden. Dafür muß man diese Schnittstelle kontrollieren können. Wie sind die SLIP-Namen den seriellen Schnittstellen zugeordnet? Der Netzwerk Code verwendet einen *ioctl* (I/O Control) Aufruf, um die serielle Schnittstelle in ein SLIP-Device 'umzuschalten'. Es gibt zwei Programme, die diese Aufgabe übernehmen: `dip` und `slattach`.

### 5.25.1 dip

`dip` (Dialup IP) ist ein intelligentes Programm, das die Übertragungsgeschwindigkeit der seriellen Schnittstelle einstellen kann, das Modem zum Wählen veranlaßt, automatisch die eingehenden Meldungen der Gegenstelle nach den notwendigen Informationen wie IP-Adresse durchsucht und die notwendigen *ioctl* Aufrufe ausführt, um die Schnittstelle in den SLIP Modus zu schalten. `dip` unterstützt eine umfangreiche Script-Sprache und kann dadurch den gesamten Login-Prozeß automatisieren.

Die Bezugsquelle ist

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

Zur Installation gehen Sie wie folgt vor:

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

<Makefile editieren>

# make install
#
```

Das Makefile nimmt die Existenz einer Gruppe *uucp* an, dies kann aber leicht z.B. in *dip* oder *SLIP* umgeändert werden.

### 5.25.2 slattach

Im Gegensatz zu `dip` ist `slattach` ein extrem einfaches Programm. Es ist einfach zu benutzen, bietet aber nicht den Komfort oder die Script-Fähigkeit von `dip`. Alles was es macht, ist, die serielle Schnittstelle als SLIP Device zu konfigurieren. Dabei setzt es voraus, daß sie alle notwendigen Informationen besitzen, und daß die Verbindung bereits aufgebaut ist, wenn es gestartet wird. `slattach` ist optimal geeignet, wenn sie eine dauerhafte Verbindung zu ihrem Server haben.

### 5.25.3 Wann benutze ich welches Programm?

`dip` bietet sich an, wenn die Verbindung zum SLIP Server über ein Modem oder eine andere temporäre Leitung aufgebaut wird. `slattach` ist eher für feste Verbindungen, ein fest installiertes Kabel etwa, oder eine gemietete Leitung, geeignet, für Fälle also, in denen keine besonderen Aktionen notwendig sind, um die Verbindung aufzubauen. Für weitere Informationen siehe den Abschnitt 'Dauerhafte SLIP Verbindungen'.

Die Konfiguration von SLIP ist bis auf ein paar kleine Ausnahmen sehr ähnlich zur Konfiguration eines Ethernet Device (siehe dort).

Zunächst unterscheiden sich SLIP Verbindungen von Ethernet Netzwerken dadurch, daß an einem SLIP-'Netzwerk' immer nur zwei Rechner beteiligt sind. Außerdem sind bei SLIP Verbindungen oft zusätzliche Maßnahmen notwendig, um die Netzverbindung zu aktivieren, wohingegen bei einer Ethernet Netzwerk die Verbindung bereits mit dem Einstecken der Kabel besteht.

Wenn Sie `dip` verwenden, wird der Verbindungsaufbau normalerweise nicht bereits beim Booten vorgenommen sondern erst zu einem späteren Zeitpunkt, wenn eine Netzverbindung benötigt wird. Es ist auch dann möglich, diesen Vorgang zu automatisieren. Falls Sie `slattach` verwenden werden Sie vermutlich lieber einen speziellen Abschnitt in der Datei `rc.inet1` einfügen wollen. Dies wird etwas später beschrieben.

Es gibt zwei unterschiedliche Arten von SLIP Servern: Solche die die Adressen dynamisch vergeben, und solche die statische Adressen verwenden. Praktisch jeder SLIP Server wird sie beim Login auffordern, ihren Benutzernamen sowie ihr Paßwort einzugeben. `dip` kann diese Loginprozedur übernehmen und automatisch durchführen.

### 5.25.4 Statische SLIP Server und `dip`.

Bei einem statischen SLIP Server bekommen Sie eine IP Adresse für ihre alleinige Verwendung zugewiesen. Bei jedem Verbindungsaufbau zum Server bekommen Sie also diese feste Adresse. Der statische SLIP Server wird also ihren Modem-Anruf entgegennehmen, die normale Login-Prozedur durchführen und dann alle Datagramme an ihre IP Adresse über diese Leitung routen. Wenn Sie Zugang zu einem solchen statischen Server haben, sollten Sie einen festen Eintrag mit ihrem Rechnernamen und der IP Adresse in der Datei `/etc/hosts` einfügen. Auch in den folgenden Dateien sollten Sie entsprechende Konfigurationsänderungen vornehmen: `rc.inet2`, `host.conf`, `resolv.conf`, `etc/HOSTNAME` sowie `rc.local`. Denken Sie auch daran, daß bei der Konfiguration von `rc.inet1` keine besonderen Befehle zur Konfiguration der SLIP Verbindung benötigt werden, dies wird zur gegebenen Zeit von `dip` erledigt. Dazu müssen ihm lediglich die notwendigen Informationen mitgeteilt werden, dann wird die Konfiguration automatisch durchgeführt, nachdem die Einwählprozedur beendet ist.

Falls Ihr SLIP Server statische Adressen verwendet, können Sie den folgenden Abschnitt überspringen und gleich den Abschnitt 'Benutzung von `dip`' lesen.

### 5.25.5 Dynamische SLIP Server und `dip`.

Ein dynamischer SLIP Server vergibt die IP Adressen zufällig aus einem Pool von vorhandenen Adressen. Es gibt also keine Garantie, das man bei jeder Verbindung eine bestimmte IP Adresse zugewiesen bekommt, und die von Ihnen bei einer Sitzung verwendete Adresse kann, nachdem Sie die Verbindung beendet haben, von einem anderen Benutzer verwendet werden. Der Administrator des SLIP Servers hat für diesen Zweck einen Pool von IP Adressen reserviert, und bei einem Verbindungsaufbau bekommen Sie die erste freie Adresse zugewiesen. Diese wird dem Anrufer nach dem Verbindungsaufbau übermittelt und ist für ihn für die Dauer der Verbindung reserviert.

Die Konfiguration verläuft hier recht ähnlich wie im Falle von statischen SLIP Servern, allerdings muß in einem Zusätzlichen Schritt die zugewiesene IP Adresse ermittelt werden, um das SLIP Device entsprechend zu konfigurieren.

Auch in diesem Fall übernimmt `dip` den schwierigen Teil, die neueren Versionen sind intelligent genug um nicht nur den Verbindungsaufbau durchzuführen sondern auch automatisch die übermittelte IP Adresse zu erkennen und damit das SLIP Device zu konfigurieren.

### 5.25.6 Die Benutzung von dip.

Wie bereits erwähnt, handelt es sich bei dip um ein mächtiges Programm, welches den aufwendigen Prozeß des Einwählens in einen SLIP Server, die Loginprozedur sowie die Konfiguration des SLIP Device vereinfachen und automatisieren kann.

Um dip zu verwenden benutzt man im Allgemeinen ein 'dip Script', das eigentlich nur aus einer Liste von Kommandos besteht, die dip versteht, und die ihm mitteilen, wie die notwendigen Aktionen durchgeführt werden sollen. Die Datei sample.dip, die Bestandteil des Paketes ist, vermittelt einen ersten Eindruck, wie das vor sich geht. dip ist ein Programm mit vielen Optionen. Sie alle hier aufzulisten wäre müßig, lesen Sie dazu bitte die Online Hilfe, die Beispieldatei sowie die Datei README des dip-Paketes.

Sie werden feststellen, daß die Beispieldatei sample.dip von einem statischen SLIP Server ausgeht, die verwendete IP Adresse also bereits bekannt sein muß. Für dynamische SLIP Server gibt es in den neueren Versionen von dip ein spezielles Kommando, mit dem man automatisch die IP Adresse aus den Antworten des Servers extrahieren kann, um damit dann das SLIP Device zu konfigurieren. Das folgende Script ist eine veränderte Version der Datei sample.dip, die mit der Version dip337j-uri.tgz ausgeliefert wird. Sie stellt vermutlich einen ausreichenden Startpunkt für alle dar, die einen dynamischen SLIP Server verwenden. Speichern sie es unter dem Namen /etc/dipscript und verändern Sie es entsprechend ihrer eigenen Konfiguration:

```
#
# sample.dip    Dialup IP connection support program.
#
#               This file (should show) shows how to use the DIP
#               This file should work for Annex type dynamic servers, if you
#               use a static address server then use the sample.dip file that
#               comes as part of the dip337-uri.tgz package.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#
main:
# Lege Namen und Adresse des Servers fest
# Mein Server heisst 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Setze die Netzmaske fuer sl0 auf 255.255.255.0
netmask 255.255.255.0
# Lege die verwendete serielle Schnittstelle und die Geschwindigkeit fest
port cua02
speed 38400

# Reset fuer das Modem und die Terminal Verbindung.
# Das verursacht fuer manche Leute Probleme!
reset

# Hinweis! "Standard" vordefinierte "errlevel" Werte sind:
# 0 - OK
# 1 - CONNECT
# 2 - ERROR
#
# Man kann sie aendern, Suchen Sie (mit grep) nach "addchat()" in *.c

# Vorbereitung zum Waehlen
send ATQ0V1E1X4\r
```

```
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# Die Verbindung wurde aufgebaut, 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:

# Login erfolgreich
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

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

# Ermitteln der vom Server zugewiesenen IP Adresse
# Dabei wird vorausgesetzt, dass der Server diese Adresse nach
# dem Umschalten in den SLIP Modus ausgibt.
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Setzen der Arbeitsparameter fuer SLIP
get $mtu 296
# Dies stellt sicher, dass ein
# "route add -net default xs4all.hacktic.nl" durchgefuehrt wird
default

# Wir sind da! Starte SLIP
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT beim Starten von sliplogin...
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 Script geht von einer dynamischen SLIP Verbindung aus. Für statische SLIP Server verwenden Sie bitte die Datei `sample.dip` aus dem Paket `dip337j-uri.tgz`.

Wenn `dip` den Befehl `get $local` erhält, durchsucht es sämtlichen eingehenden Text von der anderen Seite auf eine Zeichenkette, die wie eine IP Adresse aussieht, also Zahlen, die durch Punkte getrennt sind. Diese Veränderung wurde eingeführt, damit der Verbindungsaufbau auch für dynamische SLIP Server automatisiert werden kann.

Das obige Beispiel konfiguriert automatisch einen Default Route Eintrag über das SLIP Device. Entspricht das nicht ihren Wünschen, z.B. weil Sie außerdem noch eine Ethernet Verbindung haben, die ihre 'default' Route darstellt, entfernen Sie die Zeile `default` aus dem Script. Nachdem das Script beendet ist, können Sie mit dem Befehl `ifconfig` verifizieren, daß ein Device `sl0` existiert. Dieses Device können Sie dann mit den üblichen `ifconfig` und `route` Befehlen Ihren Wünschen entsprechend konfigurieren.

Beachten Sie auch, daß sie mit `dip` mittels des `mode` Befehles unterschiedliche Protokolle nutzen können. Das am häufigsten verwendete ist wohl *CSLIP* für SLIP mit Komprimierung. Eine solche Einstellung muß aber auf beiden Seiten identisch sein, verwenden Sie also die Einstellung ihres Servers.

Das Beispiel ist recht robust und sollte die meisten Fehler abfangen. Bei weiteren Fragen informieren Sie sich bitte über die Online Hilfe zu `dip`. Selbstverständlich kann ein solches Script auch derart erweitert werden, daß bei einem gescheiterten Einwahlversuch erneut gewählt wird, oder sogar eine andere Nummer angerufen wird usw.

### 5.25.7 Dauerhafte SLIP Verbindungen mit `slattach`.

Wenn sie zwei Rechner direkt über ein Kabel miteinander verbinden, oder in der glücklichen Lage sind, über eine gemietete Standleitung mit dem Internet verbunden zu sein, können Sie sich die aufwendige Prozedur mit `dip` ersparen. `slattach` ist ein extrem einfach zu benutzendes Programm das gerade genug Funktionalität bietet, um die Verbindung richtig zu konfigurieren.

Da es sich um eine dauernde Verbindung handelt, ist der einfachste Weg, die Befehle zur Konfiguration in der Datei `rc.inet1` einzubauen. Im Prinzip besteht diese Konfiguration lediglich darin, sicherzustellen, daß die serielle Schnittstelle mit der korrekten Geschwindigkeit betrieben und in den SLIP Modus umgeschaltet wird. Mit `slattach` erreichen sie dies mit einem einzigen Befehl. Fügen Sie einfach folgende Zeilen in ihr `rc.inet1` ein:

```

#
# Aufbau einer dauerhaften statischen SLIP Verbindung
#
# Konfiguriere /dev/cua0 fuer 19.2kbps und CSLIP
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# Ende statisches SLIP.

```

Hierbei ist:

**IPA.IPA.IPA.IPA**

Ihre IP Adresse.

**IPR.IPR.IPR.IPR**

die IP Adresse des anderen Rechners



`slattach` weist dem angegebenen Seriellen Device das erste freie SLIP Device zu, beginnend mit `s10`. Der erste Aufruf von `slattach` konfiguriert also das Device `s10`, ein weiterer Aufruf `s11` usw.

Mit `slattach` können mittels der Option `-p` eine Reihe von Protokollen eingestellt werden. Im Normalfall sind das meist SLIP oder CSLIP, je nachdem ob Komprimierung verwendet werden soll oder nicht. In jedem Fall muß aber auf beiden Seiten dieselbe Einstellung verwendet werden.

## 5.26 SLIP server.

Wenn Sie einen Rechner mit Netzwerkzugang besitzen, über den Sie anderen Nutzern die Einwahl in das Netz ermöglichen wollen, müssen Sie diesen Rechner als Server konfigurieren. Wenn Sie für die Verbindung als serielles Protokoll SLIP verwenden wollen, haben Sie drei Möglichkeiten unterschiedliche Möglichkeiten für diese Konfiguration. Ich würde den ersten Vorschlag, `sliplogin`, bevorzugen, da er am einfachsten zu realisieren und zu verstehen ist. Aber treffen Sie ihre eigene Entscheidung.

### 5.26.1 SLIP Server mit `sliplogin`.

`sliplogin` können Sie anstelle der normalen Login-Shell für Nutzer verwenden, die sich in ihren Rechner einwählen. Das Programm schaltet automatisch die serielle Verbindung in den SLIP Modus und bietet Unterstützung sowohl für statische als auch für dynamische IP Adressenvergabe.

Der Benutzer führt einen normalen Login-Prozeß durch, also Eingabe von Benutzererkennung und Paßwort, aber statt dann eine Shell vorgesetzt zu bekommen wird `sliplogin` gestartet, das in der Datei `/etc/slip.hosts` nach einem Eintrag für den anrufenden Benutzer sucht. Wird dieser gefunden, wird die Verbindung als 8bit-rein konfiguriert und über einen `ioctl` Aufruf in den SLIP Modus geschaltet. Danach startet `sliplogin` als letzten Schritt ein Script, in dem das SLIP Device mit den entsprechenden Parametern (IP Adresse, Netz Maske, Routing) konfiguriert wird. Dieses Script heißt üblicherweise `/etc/slip.login`, aber wie auch bei `getty` können sie für Benutzer, die einer besonderen Behandlung bedürfen, eigene Scripts unter dem Namen `/etc/slip.login.loginname` anlegen, die dann anstelle des Standardscriptes gestartet werden.

Es gibt drei bzw. vier Dateien, die konfiguriert werden müssen, damit `sliplogin` richtig funktioniert:

- `/etc/passwd`, für die Accounts der einwählenden Nutzer.
- `/etc/slip.hosts`, hier stehen die Nutzer-spezifischen Informationen für jeden einzelnen Einwähler.
- `/etc/slip.login`, dieses Script regelt die Routing Konfiguration für die Nutzer.
- `/etc/slip.tty`, diese Datei wird nur bei der Verwendung von *dynamischer* Adressvergabe benötigt und enthält eine Tabelle mit benutzbaren Adressen.
- `/etc/slip.logout`, hier stehen die Kommandos um die Verbindung bei Logout oder Fehlern korrekt zu beenden.

**Bezugsquellen für `sliplogin`.** Eventuell ist `sliplogin` bereits Bestandteil ihrer Linux-Distribution. Wenn nicht bekommt man es von

`sunsite.unc.edu:/pub/linux/system/Network/serial/sliplogin-2.1.1.tar.gz`

Die TAR Datei enthält Quellen, vorkompilierte Binärprogramme und die Manpage.

Um sicherzustellen daß nur autorisierte Nutzer `sliplogin` benutzen können, sollten Sie in de Datei `/etc/group` einen Eintrag wie diesen hier vorsehen:

```
..
slip::13:radio,fred
..
```

Bei der Installation von `sliplogin` wird das Makefile die Eigentumsrechte für `sliplogin` auf die Gruppe `slip` setzen. Dadurch können nur Nutzer, die in dieser Gruppe sind, das Programm ausführen. Im oben angeführten Beispiel wären das die Nutzer `radio` und `fred`.

Um die Programme im Verzeichnis `/sbin` und die Manpages in der Sektion 8 zu installieren gehen Sie folgendermassen vor:

```
# cd /usr/src
# gzip -dc ../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..Makefile editieren falls Sie keine Shadow Passworte verwenden..>
# make install
```

Falls Sie die Programme vor der Installation selber neu übersetzen wollen, fügen Sie vor dem `make install` noch ein `make clean` ein. Sollen die Programme in eine anderes Verzeichnis installiert werden, müssen Sie im Makefile die Regel `install` entsprechend editieren.

Lesen Sie bitte auch die Datei `README` die zum Paket gehört.

**Anpassung von `/etc/passwd` für SLIP Hosts.** Normalerweise richtet man auf dem für jeden Benutzer von SLIP einen speziellen Account in `/etc/passwd` ein. Eine Konvention hierbei ist es, als Benutzernamen eingroßes 'S', gefolgt vom Namen des einwahlenden Rechners, zu verwenden. Ein Rechner mit dem Namen `radio` bekommt also folgenden Eintrag:

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

Diese Konvention ist allerdings nicht zwingend, Sie können jeden beliebigen Namen verwenden, der ihnen aussagekräftig genug erscheint.

Hinweis: Der Anrufer benötigt kein besonderes Heimatverzeichnis, da er von diesem Rechner niemals eine Shell zu Gesicht bekommen wird. `/tmp` ist deshalb eine gute Wahl für diesen Zweck. Beachten Sie auch den Eintrag `/sbin/sliplogin` als Login-Shell.

**Konfiguration von `/etc/slip.hosts`** In der Datei `/etc/slip.hosts` sucht `sliplogin` nach Einträgen, die dem Namen des Anrufers entsprechen. In dieser Datei werden IP Adresse und Netzmaske festgelegt, die dem Anrufer zugewiesen werden. Das folgende Beispiel enthält Einträge für zwei Rechner, `radio` und `albert`, wobei letzterem die IP Adresse dynamisch zugewiesen wird:

```
#
Sradio 44.136.8.99 44.136.8.100 255.255.255.0 normal -1
Salbert 44.136.8.99 DYNAMIC 255.255.255.0 compressed 60
#
```

Die einzelnen Einträge sind:

1. Login-Name des Anrufers
2. IP Adresse des Servers
3. IP Adresse, die dem Anrufer zugeteilt wird. Enthält dieses Feld den Eintrag `DYNAMIC`, wird die IP Adresse basierend auf den Informationen in der Datei `/etc/slip.tty` bestimmt. **Achtung:** Das funktioniert erst ab Version 1.3 von `sliplogin`!
4. Netzmaske für den Anrufer in Dezimalpunktschreibweise, für ein Klasse-C Netz also `255.255.255.0`.
5. Verwendeter SLIP Modus, hier können Kompression sowie einige andere Besonderheiten eingestellt werden.

6. Timeout. Hier kann man einstellen, wie lange eine Verbindung unbenutzt sein darf (d.h. es werden keine Datagramme gesendet/empfangen), bevor die Verbindung automatisch unterbrochen wird. Ein negativer Wert verhindert das automatische Unterbrechen.

7. Optionale Argumente

Hinweise: In den Feldern 2 und 3 können sowohl Rechnernamen als auch IP Adressen in Dezimalpunktschreibweise stehen. Wenn Sie Rechnernamen verwenden, müssen diese allerdings auflösbar sein, d.h. der Server muß in der Lage sein, die zu dem Namen gehörende IP Adresse herauszufinden. Überprüfen können Sie dies z.B. durch ein `telnet` auf diesen Rechnernamen. Bekommen sie dann die Meldung `Trying nnn.nnn.nnn...`, hat ihr Rechner den Namen einwandfrei aufgelöst. Bekommen Sie hingegen die Meldung `Unknown host`, ist der Versuch fehlgeschlagen. Dann verwenden Sie entweder direkt die IP Adresse, oder stellen Sie ihr Name Resolving so ein, daß der Name gefunden wird (siehe dazu den Abschnitt 'Konfiguration des Name Resolver').

Die am häufigsten verwendeten Einstellungen für den SLIP Modus sind

#### **normal**

für normales, unkomprimiertes SLIP.

#### **compressed**

um van Jacobsen Header Kompression (cSLIP) zu aktivieren.

Die beiden Optionen schließen sich natürlich wechselseitig aus. Für weitere Informationen lesen Sie bitte die Online-Hilfe.

**Konfiguration der Datei `/etc/slip.login`.** Hat `sliplogin` einen passenden Eintrag in `/etc/slip.hosts` gefunden, wird es als nächstes Versuchen, das Script `/etc/slip.login` zu starten, um das SLIP Interface mit den notwendigen Parametern IP Adresse und Netzmaske zu konfigurieren.

Die mit dem Paket gelieferte Beispieldatei sieht folgendermassen aus:

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generische login Datei fuer eine SLIP Verbindung.  sliplogin
# ruft das Script mit ff. Parametern auf:
#   $1      $2      $3      $4, $5, $6 ...
#   SLIPunit ttyspeed  pid  die Argumente aus dem Eintrag in slip.host
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Sie werden feststellen, daß dieses Script ganz einfach nur die Befehle `ifconfig` und `route` verwendet, um das SLIP Device zu konfigurieren, genau wie das auch bei der Verwendung von `slattach` der Fall wäre.

Beachten Sie auch die Verwendung von *Proxy ARP*. Damit wird sichergestellt, daß andere Rechner, die am selben Ethernet Netzwerk wie der Server angeschlossen sind, den einwählenden Rechner erreichen können. Ist ihr Server nicht an ein Ethernet Netz angeschlossen, können Sie diese letzte Zeile ganz auslassen.

**Konfiguration von `/etc/slip.logout`.** Falls die Verbindung zusammenbricht sollten Sie sicherstellen, daß die serielle Schnittstelle in ihren Normalzustand zurückversetzt wird, damit der nächste Anrufer sich ganz normal einloggen kann. Dies erreichen sie mit dem Script `/etc/slip.logout`. Es hat ein sehr einfaches Format und wird mit denselben Parametern wie `/etc/slip.login` aufgerufen (auch wenn davon nur ein paar verwendet werden...)

```
#!/bin/sh -
#
#           slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Alles was es macht, ist das Interface herunterzufahren, wodurch automatisch auch die vorher angelegte Route gelöscht wird. Den hier ebenfalls enthaltenen `arp` Aufruf können Sie auch wieder löschen, falls Sie nicht an ein Ethernet Netzwerk angeschlossen sind.

**Konfiguration von `/etc/slip.tty`.** Falls Sie dynamische IP Adressen verwenden (also mindestens einen der Rechner mit dem Eintrag `DYNAMIC` konfiguriert haben), dann müssen Sie auch die Datei `/etc/slip.tty` konfigurieren, indem Sie dort alle zur Auswahl stehenden Adressen auflisten. Sie benötigen diese Datei aber nur für die dynamische Vergabe von IP Adressen.

Die Datei ist eine Tabelle, die die `tty`-Devices auflistet, über die `SLIP` Verbindungen eingehen können, und die IP Adresse, die einem Anrufer auf dem jeweiligen Port zugewiesen wird:

```
# 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
#
```

Das vorstehende Beispiel legt also fest daß all denjenigen Anrufern, die sich über den Port `/dev/ttyS0` einwählen und in dem entsprechenden Feld in der Datei `/etc/slip.hosts` den Eintrag `DYNAMIC` haben, die IP Adresse `192.168.0.100` zugewiesen bekommen.

Dadurch benötigt man nur eine Adresse je zur Verfügung stehenden Port und kann so die Anzahl der belegten Adressen klein halten.

### 5.26.2 SLIP Server mit `dip`.

Zu Beginn ein Hinweis: Einige der in diesem Abschnitt gegebenen Informationen entstammen der Hilfe-Seite von `dip`, in der ebenfalls eine kurze Anleitung gegeben wird, wie Linux als `SLIP` Server konfiguriert werden kann. Alle Angaben hier beziehen sich auf die Version `dip3370-uri.tgz` und gelten nicht automatisch für andere Versionen dieses Paketes.

`dip` hat einen speziellen Eingabemodus, in dem es für denjenigen, der es gestartet hat, automatisch alle notwendigen Informationen aus der Datei `/etc/diphosts` zusammensucht, um die serielle Verbindung zu konfigurieren und in den `SLIP` Modus zu schalten. Dieser besondere Modus wird aktiviert, wenn das Programm unter dem Namen `diplogin` gestartet wird. Um `dip` auf eine Server zu verwenden müssen Sie also lediglich besondere Accounts einrichten, die `diplogin` als Login-Shell verwenden.

Dafür muß zunächst ein symbolischer Link angelegt werden:

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

Dann müssen entsprechende Einträge in `/etc/passwd` und `/etc/diphosts` vorgenommen werden.

Für jeden Benutzer wird - wie auch bei `sliplogin` - ein Account angelegt, Konvention ist auch hier, den Nutzernamen mit einem großen 'S' zu beginnen. Das sieht dann etwa so aus:

```
Sfredm:ij/SMxiTLGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^          ^^          ^^  ^^  ^^  ^^  ^^
|           |           |   |   |   |   \__ diplogin als Login Shell
|           |           |   |   |   |   \_____ Heimatverzeichnis
|           |           |   |   |   |   \_____ Voller Nutzername
|           |           |   |   |   |   \_____ User Group ID
|           |           |   |   |   |   \_____ User ID
|           |           |   |   |   |   \_____ Verschlussetes Passwort
|           |           |   |   |   |   \_____ Slip User Login Name
```

Der Login wird wie gewöhnlich vom Programm `login(1)` abgewickelt. Ist alles in Ordnung, wird das Programm `diplogin` gestartet. `dip`, mit dem Namen `diplogin` aufgerufen, weiß dann automatisch, daß es als Login-Shell benutzt wird. Als erstes ruft es dann die Funktion `getuid()` auf um die Benutzer ID desjenigen herauszufinden, der das Programm gestartet hat. Danach sucht es in der Datei `/etc/diphosts` nach dem ersten Eintrag, der entweder der Benutzer-ID oder aber dem Namen des `tty` entspricht, über den die Verbindung aufgebaut wurde, und führt dementsprechend die Konfiguration durch. Durch die Entscheidung, einem Nutzer entweder einen Eintrag für seine ID zuzuweisen, oder die Standardeinstellung für das `tty` zu verwenden können einfach statische und dynamische Adressen parallel verwendet werden.

`dip` fügt in diesem Modus automatisch einen Eintrag für Proxy-ARP durch, dies muß also nicht von Hand geschehen.

**Die Konfiguration von `/etc/diphosts`.** Die Datei `/etc/diphosts` wird von `dip` verwendet, um voreingestellte Konfigurationen für unterschiedliche Rechner zu speichern. Dabei kann es sich um Rechner handeln, die sich in ihren Rechner einwählen, aber auch um solche, in die Sie sich mit ihrem Rechner einwählen.

Das allgemeine Format der Einträge in `/etc/diphosts` 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 einzelnen Einträge bedeuten:

1. *Login Name*: wie er von `getpwuid(getuid())` zurückgeliefert wird, oder Name des `tty`.
2. *unbenutzt*: zur Kompatibilität mit `passwd`.
3. *Remote Adresse*: IP Adresse des anrufenden Rechners, entweder als Name oder in Dezimalschreibweise.
4. *Lokale Adresse*: IP Adresse des lokalen Rechners, entweder als Name oder in Dezimalschreibweise.
5. *Netzmaske*: in Dezimalschreibweise.
6. *Kommentar*: Beliebiger Eintrag.
7. *Protokoll*: SLIP, CSLIP usw.
8. *MTU*: Zahl.

Der untere der beiden Beispielenträge legt also z.B. fest, daß ein Anrufer auf `ttyS1` die (dynamische) Adresse `145.71.34.3` zugewiesen bekommt und die Verbindung mit Komprimierung (CSLIP) und einer MTU von 296 konfiguriert wird.

Alle Nutzer, die eine *statische* IP Adresse zugewiesen bekommen sollen, müssen einen Eintrag unter ihrem Login-Namen in `/etc/diphosts` haben. Für andere Anrufer, denen die IP Adresse dynamisch zugewiesen werden soll, muß ein Eintrag für die in Frage kommenden *tty* Ports vorhanden sein. Es sollte auf jeden Fall für jeden vorhandenen Port ein Eintrag vorhanden sein um sicherzustellen, daß ein Anrufer in jedem Fall eine gültige Konfiguration vorfindet.

Wenn sich nun ein Benutzer einlogged, wird er ganz normal nach Name und Paßwort gefragt. Hier muß er seinen SLIP Login-Namen und das zugehörige Paßwort eingeben. Verläuft alles normal wird der Benutzer keinerlei zusätzliche Meldungen bekommen, er sollte dann einfach die Verbindung in den SLIP Modus schalten, dann sollte er eine Verbindung mit den Parametern aus `diphosts` aufbauen können.

### 5.26.3 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, die Datei 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.

## 5.27 Unterstützung für STRIP (Starmode Radio IP)

Die Device Namen für STRIP sind ??? usw.

**Optionen beim Kernel kompilieren:**

```
Network device support --->
  [*] Network device support
  ....
  [*] Radio network interfaces
  < > STRIP (Metricom starmode radio IP)
```

Das STRIP Protokoll wurde speziell für eine besondere Art von Funk-Modems entwickelt, die in einem Forschungsprojekt der Universität Stanford mit dem Namen MosquitoNet Project verwendet werden:

```
http://mosquitonet.Stanford.EDU/mosquitonet.html
```

Sie finden dort eine Menge interessanter Informationen - selbst wenn Sie nicht an dem Projekt selber interessiert sind.

Die Metricom Sender werden an die serielle Schnittstelle angeschlossen, verwenden verteilte Wellenlängenbereiche und können typischerweise etwa 100kbps übertragen. Informationen über diese Sender finden sie auf dem Metricom Web Server ([www.metricom.com](http://www.metricom.com)).

Die normalen Netzwerkprogramme unterstützen dieses Protokoll derzeit nicht, sie müssen sich also speziell angepaßte Versionen vom MosquitoNet Webserver beschaffen. Genauere Informationen, welche Software Sie benötigen, finden sie auf der MosquitoNet STRIP Page:

```
http://mosquitonet.Stanford.EDU/strip.html
```

Eine kurze Zusammenfassung der Konfiguration: Sie verwenden eine modifizierte Version des Programmes `slattach`, um die serielle Verbindung in den STRIP Modus zu schalten, und konfigurieren die neuen Devices dann wie ein normales Ethernet Device. Einziger wichtiger Unterschied: STRIP unterstützt kein ARP, die ARP Einträge für alle Rechner eines Sub-Netzwerkes müssen also von Hand vorgenommen werden.

## 5.28 Token Ring

Die Namen der Token Ring Devices sind `tr0`, `tr1` usw. Token Ring ist ein Standard LAN Protokoll von IBM, bei dem Kollisionen von Datagrammen dadurch vermieden werden, daß jeweils immer nur ein Rechner des LAN das Recht hat, Daten zu übertragen. Auf dem LAN wird ein 'Token' vergeben, das zu einem beliebigen Zeitpunkt immer nur ein Rechner haben kann. Nur dieser Rechner ist befugt, zu senden. Sind die Daten übertragen, wird das Token an den nächsten Rechner weitergegeben. Das Token wandert also zwischen allen aktiven Rechnern herum, daher der Name 'Token Ring'.

### Optionen beim Kernel kompilieren:

```
Network device support --->
  [*] Network device support
  ....
  [*] Token Ring driver support
  < > IBM Tropic chipset based adaptor support
```

Die Konfiguration eines Token Ring Device ist bis auf die anderen Devicenamen identisch zur Konfiguration eines Ethernet Device.

## 5.29 X.25

X.25 ist ein Packet Switching Protokoll, das durch die C.C.I.T.T. festgelegt wurde, einer Basis von Standards, die von Telefongesellschaften in den meisten Teilen der Welt anerkannt sind. An einer Implementation von X.25 und LAPB wird derzeit gearbeitet, die jeweils aktuelle Version ist Bestandteil der Entwickler-Kernels 2.1.\*.

Jonathon Naylor ([jn@cs.nott.ac.uk](mailto:jn@cs.nott.ac.uk)) leitet die Entwicklung, es wurde eine Mailing Liste angelegt, über die Diskussionen zum Thema X.25 unter Linux geführt werden. Um sie zu abonnieren schicken sie eine Mail an [majordomo@vger.rutgers.edu](mailto:majordomo@vger.rutgers.edu) mit dem Text "subscribe linux-x25" als Inhalt der Mail.

Erste Versionen der Konfigurationsprogramme bekommen Sie per FTP von:

```
ftp.cs.nott.ac.uk:/jsn/
```

## 5.30 WaveLan Karten

Die Device Namen für WaveLan sind `eth0`, `eth1` usw.

### Optionen beim Kernel kompilieren:

```

Network device support --->
  [*] Network device support
  ....
  [*] Radio network interfaces
  ....
  <*> WaveLAN support

```

WaveLAN Karten sind für Kabellose Verbindungen und verwenden Multifrequenztechnik. Die Karten verhalten sich praktisch wie Ethernet-Karten und werden genauso konfiguriert.

Informationen über diese Karten bekommen Sie von Wavelan (<http://www.wavelan.com>).

## 6 Kabel und Verkabelung

Wer mit einem LötKolben umgehen kann möchte sich vielleicht ein Kabel basteln, um zwei Linux Rechner zu verbinden. Die folgenden Diagramme sollten Sie dabei unterstützen.

### 6.1 Ein Serielles NULL Modem Kabel

Nicht alle NULL Modem Kabel sind gleich. Viele dieser Kabel machen nicht mehr als dem Rechner vorzugaukeln daß die notwendigen Signale vorhanden sind und vertauschen Transmit und Receive. Das ist im Prinzip OK, bedeutet aber, daß Sie Software Flow Control (XON/XOFF) verwenden müssen, was weniger effizient ist als Hardware Flow Control. Das folgende Kabel bietet die bestmögliche Signalverbindung zwischen Rechnern und erlaubt die Verwendung von Hardware Flow Control (RTS/CTS).

Pin Name	Pin		Pin
Tx Data	2	-----	3
Rx Data	3	-----	2
RTS	4	-----	5
CTS	5	-----	4
Ground	7	-----	7
DTR	20	- \-----	8
DSR	6	- /	
RLSD/DCD	8	----- /-	20
		\-	6

### 6.2 Kabel für die Parallelschnittstelle (PLIP)

Wenn Sie zur Verbindung zweier Rechner das PLIP Protokoll nutzen wollen, sollten sie folgende Verkabelung wählen, die unabhängig von der Art der verwendeten Parallelschnittstelle ist.

Pin Name	pin		pin
STROBE	1*		
D0->ERROR	2	-----	15
D1->SLCT	3	-----	13
D2->PAPOUT	4	-----	12
D3->ACK	5	-----	10
D4->BUSY	6	-----	11
D5	7*		
D6	8*		
D7	9*		
ACK->D3	10	-----	5
BUSY->D4	11	-----	6



PAPOUT->D2	12	-----	4
SLCT->D1	13	-----	3
FEED	14*		
ERROR->D0	15	-----	2
INIT	16*		
SLCTIN	17*		
GROUND	25	-----	25

Hinweise:

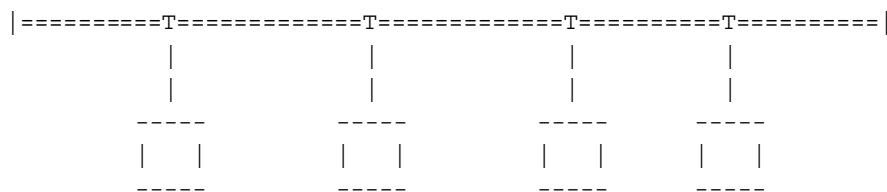
- Die mit einem Stern \* gekennzeichneten Pins dürfen nicht verbunden werden.
- Geschirmte Kabel sollten **nur auf einer Seite** mit dem Metall des Steckers verbunden werden.
- **Vorsicht: Ein falsch verdrahtetes PLIP-Kabel kann ihre Controller Karte zerstören!** Seien Sie sehr vorsichtig, und überprüfen sie jede Verbindung doppelt um unnötigen Ärger zu vermeiden.

### 6.3 (Thin) Ethernet Verkabelung (10base2)

10base2 ist ein Ethernet Standard, der die Verwendung von 52 Ohm Koaxialkabel mit einem Durchmesser von etwa 5mm vorschreibt. Bei der Verbindung von Rechnern mit dieser Art von Kabel sind einige wichtige Regeln zu beachten.

Das erste ist, daß an **beiden Enden** des Kabels Terminatoren angebracht werden müssen. Ein Terminator ist ein 52 Ohm Widerstand der sicherstellt, daß ein Signal am Ende des Kabels nicht reflektiert wird. Ohne einen solchen Endwiderstand arbeitet das Netzwerk unzuverlässig oder gar nicht.

Normalerweise werden T-Stücke verwendet, um Rechner anzuschließen, das Netzwerk sieht also etwa so aus:



die Zeichen | an beiden Enden stellen die Terminatoren dar, ==== ist ein Stück Koaxial-Kabel mit BNC-Steckern, T das erwähnte T-Stück. Die Kabellänge zwischen T-Stück und der Ethernetkarte muß äußerst kurz gehalten werden, normalerweise sollte es direkt am Ausgang der Ethernetkarte angebracht werden.

## 7 Glossar der im Text verwendeten Ausdrücke

### ARP

Ein Akronym für *Address Resolution Protocol*. Es beschreibt, wie ein vernetzter Rechner einer IP Adresse eine Hardware Adresse zuordnet.

### ATM

Ein Akronym für *Asynchronous Transfer Mode*. Ein ATM Netzwerk verpackt Daten in Blocks einer vorgegebenen Größe, die effizient von einem zum anderen Punkt übertragen werden kann.

### Klient

Damit bezeichnet man meist eine Software auf der Nutzer-Seite eines Systems. Es gibt da aber Ausnahmen, z.B. ist das X Window System ein Server auf der Nutzer-Seite, und das X-Programm ein Klient auf dem anderen (remote) Rechner, der die Dienste des (lokalen) Servers in Anspruch nimmt. Bei einem *Peer-to-Peer* Netzwerk wie SLIP oder PPP ist der Klient diejenige Seite, die die Verbindung initiiert, die angerufene Seite bezeichnet man als Server.

**Datagramm**

Ein Datagramm ist ein einzelnes Datenpaket, bestehend aus Daten und einem Header, der die Adressen enthält. Basiseinheit der Übertragung über ein IP Netzwerk, wird manchmal auch als 'Paket' bezeichnet.

**DLCI**

*Data Link Connection Identifier*, bezeichnet eine eindeutige Point-to-Point Verbindung über ein Frame Relay Netzwerk. DLCI's werden normalerweise vom Provider festgelegt.

**Frame Relay**

Eine Netzwerktechnologie die insbesondere für Datenverkehr optimiert ist, der in Spitzen auftritt. Die Nettwerkkosten werden reduziert, indem sich mehrere Nutzer die Bandbreite einer Verbindung teilen. Dabei wird davon ausgegangen daß die Hauptnutzungszeiten der verschiedenen Teilnehmer sich nicht überschneiden.

**Hardware Adresse**

Eine Zahl, die einen Rechner in einem physikalischen Netzwerk eindeutig auf Hardware-Zugriffsebene identifiziert. Beispiele hierfür sind die Ethernet-Adresse oder die AX.25 Adresse.

**ISDN**

Ein Akronym für *Integrated Services Dedicated Network*. Bietet einen standardisierten Weg für Telefongesellschaften, Sprache oder Daten zum Endkunden zu übertragen.

**ISP**

Ein Akronym für Internet Service Provider - Anbieter von Internet Diensten. Organisationen oder Firmen, die anderen Personen Anschluß an das Internet anbieten.

**IP Adresse**

Eine Nummer, die einen Rechner in einem IP Netzwerk eindeutig identifiziert. Die Adressen sind 4 Byte lang und werden für gewöhnlich in der Dezimalpunktschreibweise dargestellt, bei der jedes Byte als Dezimalzahl (0-255), durch Punkte getrennt, aufgeschrieben wird.

**MSS**

Ein Akronym für *Maximum Segment Size* . Die maximale Größe eines Datenpaketes, das auf einmal übertragen werden kann. Um lokale Fragmentation zu vermeiden sollte gelten  $MSS=MTU-IP\_Header$ .

**MTU**

Ein Akronym für *Maximum Transmission Unit*. Die Maximale Größe eines Datagrammes, die über ein IP Interface übertragen werden kann, ohne in Teilstücke zerlegt werden zu müssen. Die MTU sollte größer sein als das größte Datenpaket, das man unfragmentiert übertragen will (da noch der IP-Header hinzukommt). Das verhindert eine Fragmentierung allerdings nur lokal, da andere Rechner auf der Verbindung möglicherweise kleinere Werte für die MTU verwenden und die Datenpakete dann dort fragmentiert werden. Typische Werte sind 1500 für ein Ethernet Interface und 576 für ein SLIP Interface.

**Route**

Der Weg, den ein Datenpaket vom Startrechner zum Zielrechner nimmt.

**Server**

Normalerweise eine Software auf der für den Nutzer nichtlokalen Seite einer Verbindung. Ein Server bietet einen Dienst für einen oder mehrere Klienten an. Beispiele sind *FTP*, *NFS* oder *DNS*. Bei einem Peer-to-Peer Netzwerk bezeichnet der Server den angerufenen Rechner, der anrufende Rechner ist der Klient.

**Window**

Die größte Datenmenge, die der Empfänger zu jedem beliebigen Zeitpunkt empfangen kann.

## 8 Linux für einen ISP ?

Wenn Sie Linux verwenden wollen, um Netzwerkdienste anzubieten, sollten sie einen Blick auf die Linux ISP Homepage

`http://www.anime.net/linuxisp/`

werfen. Sie finden dort viele Verweise auf hilfreiche Informationen zu diesem Thema.