

**Free!**

# Linux im Netzwerk



OSI - der Aufbau  
TCP/IP - das Protokoll  
DNS - die Adressierung  
Firewall - die Sicherheit

## Acrobat Reader: Wie ...

**F5/F6** öffnet/schließt die Ansicht **Lesezeichen**

**Strg+F** sucht

**Im Menü Ansicht stellst du ein, wie die Datei gezeigt wird**

**STRG+0** = Ganze Seite **STRG+1** = Originalgrösse **STRG+2** = Fensterbreite

Im selben Menü kannst du folgendes einstellen:: **Einzelne Seite**, **Fortlaufend** oder **Fortlaufend - Doppelseiten** .. Probiere es aus, um die Unterschiede zu sehen.

### Navigation

**Pfeil Links/Rechts**: eine Seite vor/zurück

**Alt+ Pfeil Links/Rechts**: Wie im Browser: Vorwärts/Zurück

**Strg++** vergrößert und **Strg+-** verkleinert

## Bestellung und Vertrieb für den Buchhandel

Bonner Pressevertrieb, Postfach 3920, D-49029 Osnabrück

Tel.: +49 (0)541 33145-20

Fax: +49 (0)541 33145-33

bestellung@knowware.de

www.knowware.de/bestellen

## Autoren gesucht

Der KnowWare-Verlag sucht ständig neue Autoren. Hast du ein Thema, daß dir unter den Fingern brennt? - ein Thema, das du anderen Leuten leicht verständlich erklären kannst?

Schicke uns einfach ein paar Beispielseiten und ein vorläufiges Inhaltsverzeichnis an folgende Adresse:

lektorat@knowware.de

Wir werden uns deinen Vorschlag ansehen und dir so schnell wie möglich eine Antwort senden.

<b>Vorwort des Verlags .....</b>	<b>4</b>	<b>Die Nutzung von</b>	<b>Web - Proxy, DNS-Server,</b>
<b>Grundlagen der</b>		<b>Netzwerken .....</b>	<b>Mailserver .....</b>
<b>Netzwerktechnik.....</b>	<b>5</b>	Terminaldienste .....	40
Das OSI-Modell .....	5	Telnet .....	Web-Proxydienste.....
Netzwerktopologien .....	7	r tools – rsh und rlogin.....	40
Bustopologie .....	7	s Tools – ssh und slogin.....	DNS-Server .....
Sterntopologie .....	7		40
Ringtopologie.....	7		Wie DNS funktioniert .....
Baumtopologie .....	7		40
Vernetzungsarten und		<b>Die Nutzung von</b>	BIND - die
Geschwindigkeiten .....	8	<b>Dateidiensten .....</b>	Standardsoftware .....
10 - Base - 2 .....	8	File Transfer Protocol - ftp ...	42
10 - Base - T .....	8	rcp - Dateiübertragung mit	Mailserver .....
100 - Base - T.....	8	r-Tools.....	46
Weitere Typen.....	8	scp - Dateien sicher über	Dienste für
Protokolle im Netzwerk .....	8	das Netzwerk kopieren.....	Windowsclients - Samba ....
NetBIOS.....	8	nfs – der transparente	49
IPX .....	8	Zugriff auf das	Basiskonfiguration von
TCP/IP.....	9	Dateisystem über das	Samba .....
Unterschiedliche Arten von		Netz .....	49
Netzwerken.....	9	smbmount – der Zugriff	<b>Sicherheit in Linuxnetzen... 52</b>
Peer-to-Peer-Netzwerke .....	9	auf Windowsfreigaben	Was man nicht anbieten
Client-Server-Netzwerke.....	9	im Netzwerk.....	will, braucht nicht zu
Mischumgebungen .....	9	Mail, Netnews und	laufen.....
<b>Grundlagen des TCP/IP-</b>		<b>Webdienste .....</b>	52
<b>Protokolls.....</b>	<b>10</b>	Electronic Mail .....	Dienste reglementieren.....
Funktion und Aufbau .....	10	Usenet Netnews .....	52
Adressierung im TCP/IP-		Webdienste .....	Der Einsatz von Firewalls.....
Netzwerk .....	10	Zeit- und Datumsdienste .....	54
Ports - Die Services im		rdate .....	Paketfiltering .....
Netzwerk .....	12	xntp .....	54
<b>Einrichten eines IP-</b>		<b>Die Betreuung von</b>	Masquerading oder NAT.....
<b>Netzwerks unter Linux .....</b>	<b>13</b>	<b>Netzwerkdiensten .....</b>	54
Test des IP-Netzwerks .....	15	inetd .....	Application Level
PING - Testen der		Die Betreuung von	Gateways .....
Verbindung und des		Terminaldiensten .....	54
TCP/IP Protokoll Stacks....	15	Telnet .....	Firewalltechniken unter
Fehlerquellen und		rsh & rlogin.....	Linux .....
Diagnosewerkzeuge.....	16	ssh .....	55
		Dateidienste .....	IP Filter unter Linux.....
		File Transfer Protocol -	55
		FTP.....	IP Chains anwenden.....
		nfs – das	55
		Netzwerkdateisystem .....	
		32	
		Samba – Dateidienste für	
		Windowsclients.....	
		32	
		Webdienste.....	
		33	

## Vorwort des Verlags

Das Linux-Betriebssystem wurde praktisch seit seinem Entstehen über das Internet distribuiert und entwickelt. Von Anfang an hat dieses System daher großen Wert auf seine Netzwerkfähigkeiten gelegt – so großen Wert, dass zahlreiche Server im Internet unter Linux laufen.

Dieses Heft mag in manchen Passagen etwas komplizierter erscheinen als viele bisherige KnowWare-Hefte – das liegt aber an seinem Inhalt. Der Betrieb eines Servers ist nun einmal nicht ganz so einfach wie die Arbeit an einem Einzelplatz-Rechner.

Dafür erhält man aber hier etliche Informationen, die einen guten Hintergrund bieten – auch wenn man sich eher allgemein für Netzwerktechnologie interessiert, ohne selber an das Betreiben eines Servers zu denken.

Das Heft bietet eine Einführung in Grundlagen, Aufbau, Anwendung und Betrieb eines Netzwerks unter Linux. Logischerweise konzentrieren wir uns dabei auf das TCP/IP-Protokoll, das Netzprotokoll im Internet.

Im ersten Hauptabschnitt betrachten wir die Grundlagen der Netzwerktechnologie:

- dem *vertikalen* Aufbau eines Netzwerks, also den verschiedenen Schichten, deren Zusammenwirken das Funktionieren eines Netzwerks ermöglicht
- dem *horizontalen* Aufbau, der Topologie, also der Art und Weise, wie die verschiedenen Computer hinter- oder nebeneinander geschaltet werden
- den Vernetzungsarten
- den Netzwerkprotokollen
- sowie der Rolle als Client oder Server, die der einzelne Computer spielt.

Der zweite Hauptabschnitt befasst sich mit dem TCP/IP-Protokoll, und im dritten Hauptabschnitt richten wir unser eigenes Netzwerk ein und testen es.

Der vierte Hauptabschnitt beschreibt, wie man ein solches Netzwerk als Client benutzt; der fünfte, wie man es als Server betreibt. Unter diesen Abschnitten sehen wir uns auch diverse Dienste wie Email, ftp und World Wide Web aus beiden Perspektiven an.

Der sechste Hauptabschnitt beschreibt ein Netzwerk aus der Perspektive eines Internet-Servers. Hier finden sich grundlegende Informationen über Adressenverteilung, Proxydienste und dergleichen mehr.

Schließlich befasst sich der siebente und letzte Hauptabschnitt mit Sicherheitsproblemen im Netzwerk – und wie man ihnen vorbeugen kann.

Ein Sachwortregister am Ende des Heftes hilft bei der Suche nach bestimmten Themen – bei einem Heft diesen Charakters ist seine Benutzung sehr empfehlenswert!

Der Autor Achim Schmidt ist erreichbar unter folgender Adresse:

[Achim.Schmidt@Wusel.DE](mailto:Achim.Schmidt@Wusel.DE)

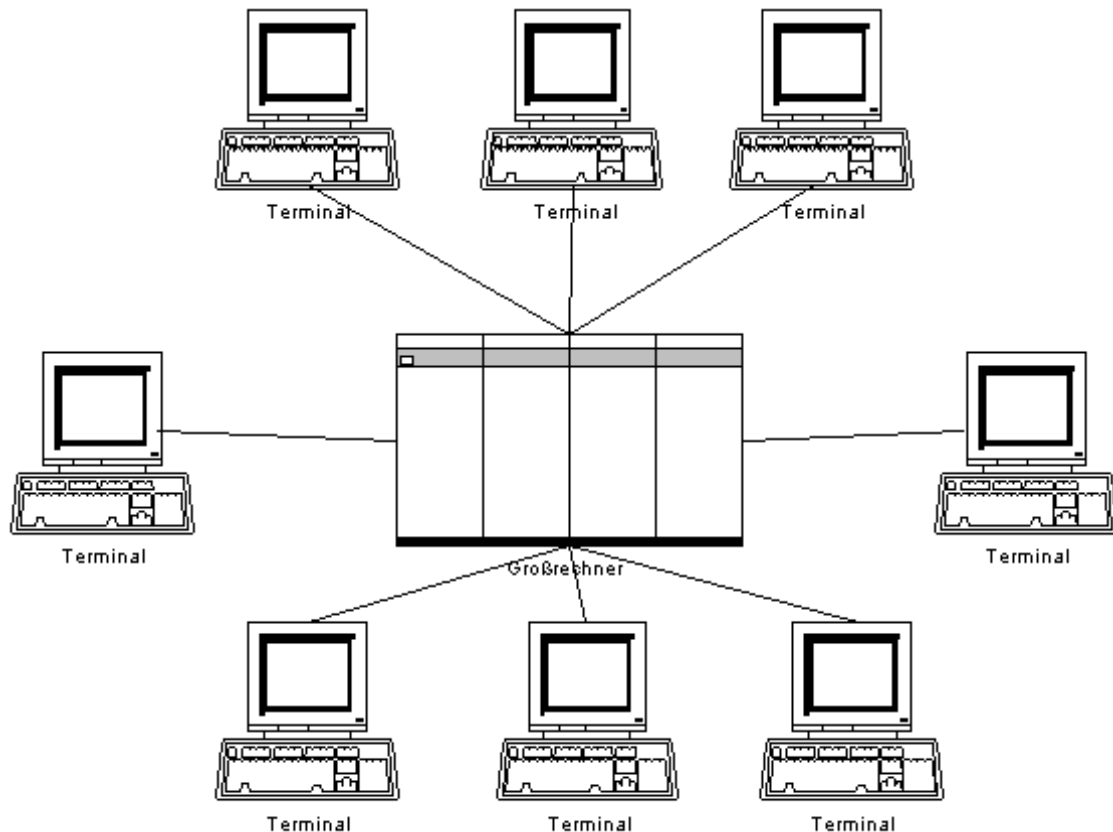
Viel Spaß bei der Lektüre,

der KnowWare-Verlag

## Grundlagen der Netzwerktechnik

Die Netzwerktechnik hat ihren Grundstein in den Grossrechneranlagen vergangener Zeiten. Damals war Vernetzung noch eine relativ simple Sache.

Ein Netzwerk bestand aus einem Großrechner-system, an das alle Terminals mittels serieller Leitungen direkt angeschlossen waren:

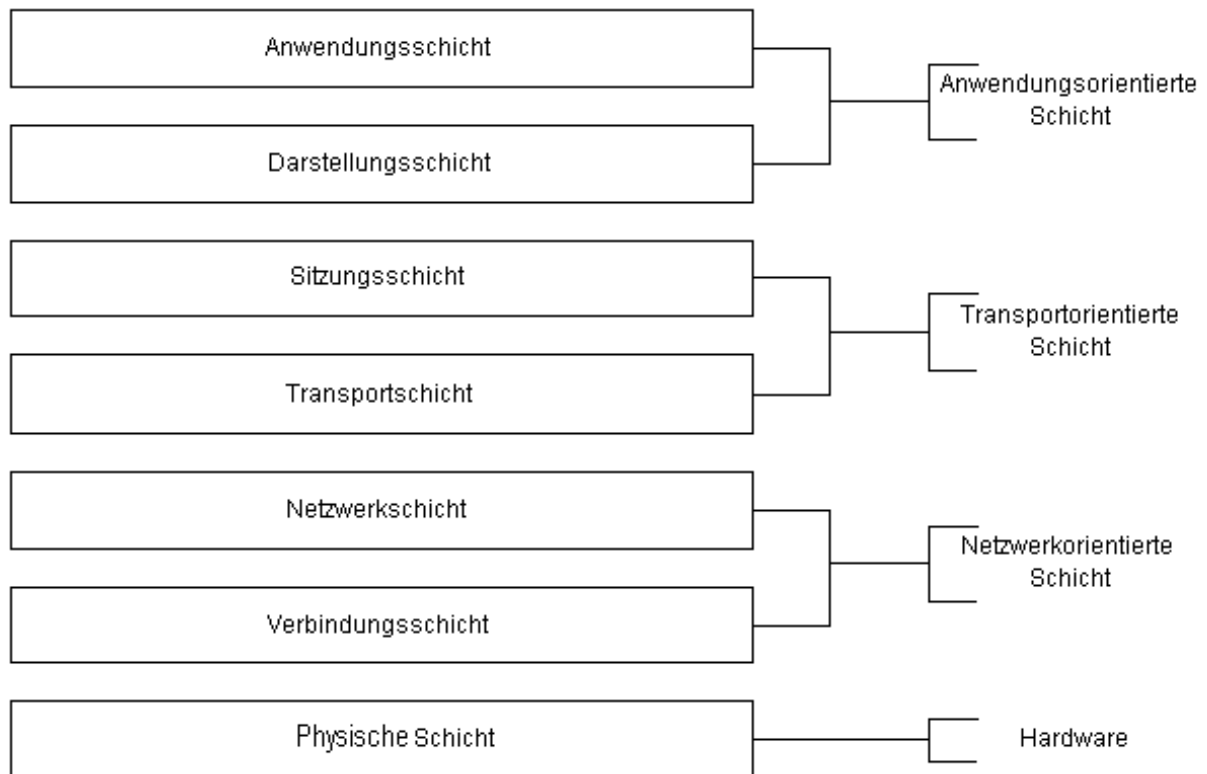


Mit dem Einzug der Mikroprozessortechnik seit 1971 wurden die Endgeräte zunehmend intelligenter, was der Netzwerktechnik ganz neue Perspektiven eröffnete. Durch die neuen Technologien wurde es allerdings nötig, unterschiedliche Geräte in einem gemeinsamen Netzwerk zu verbinden. So vielfältige Anforderungen konnten kaum durch einen Hersteller erfüllt werden. Also mußte die Kommunikation in Netzwerken standardisiert werden. 1978 wurde daher das *OSI* (Open System Interconnection)-Modell der *ISO* (International Standardisation Organisation) definiert, ein Referenzmodell für die Kommunikation innerhalb offener Systeme. Dieses Modell bot die Basis für einen gemeinsamen Standard. Damit war die Möglichkeit geschaffen, einen transparenten Datenaustausch zwischen den verschiedensten Systemen zu realisieren.

Leider läßt eine vollständige Umsetzung dieses Standards allerdings bis heute auf sich warten, da viele Hersteller sich immer noch ihrer eigenen Lösungen bedienen.

### Das OSI-Modell

Das OSI-Modell basiert auf sieben Schichten. Jede Ebene oder jedes Layer definiert eindeutig einen Teil der Kommunikation. Die Schichten sind hierarchisch angeordnet, eine höhere Schicht baut also auf der jeweils niederen auf. Allerdings handelt es sich dabei nicht etwa um einen Implementierungsleitfaden, sondern um ein Modell. So kann man einige wichtige Netzwerkprotokolle auf lediglich vier Schichten abbilden. Dennoch eignet sich dieses Modell hervorragend, um die Kommunikation in modernen Netzwerken zu erklären.



### 1. Physische Schicht (Physical Layer)

Auf der untersten Ebene des Modells werden die Spezifikationen des physischen Übertragungsmediums geregelt, also der Übertragungswege via Kabel, Funk oder Infrarot. Hier werden die Eigenschaften dieser Medien definiert.

### 2. Verbindungsschicht (Data Link Layer)

Diese Ebene ist die unterste Ebene der gesicherten Kommunikation. Es wird hier die Verbindung zum physischen Verbindungsmedium geregelt.

Der Data-Link Layer unterscheidet erneut zwei Unterebenen. Die *Media Access Control* oder *MAC*) genannte Ebene ist die untere Schicht, die ein Datenpaket von der höher gelegenen *Logical Link Control*- oder *LLC*-Schicht übernimmt und auf das Netzwerk weiterleitet.

### 3. Netzwerkschicht (Network Layer)

Auf dieser Ebene werden die betriebssystem-spezifischen Netzwerkprotokolle definiert. Typische Netzwerkprotokolle dieser Schicht sind z.B. *IP* oder *IPX*. Ausserdem wird innerhalb dieses Layers das evtl. nötige *Routing* realisiert.

### 4. Transportschicht (Transport Layer)

Die vierte Ebene des Modells stellt den kommunizierenden Anwendungen eine gesicherte und transparente Punkt-zu-Punkt-Verbindung zur Datenübertragung bereit. Des weiteren hat sie eine verbindende Funktion zwischen den unteren und oberen drei Schichten.

### 5. Sitzungsschicht (Session Layer)

Diese auch als *Verbindungssteuerungsschicht* bezeichnete Ebene hat die systemunabhängige Kontrolle des durch Schicht 4 bereitgestellten logischen Kanals zur Aufgabe.

### 6. Darstellungsschicht (Presentation Layer)

Aufgabe dieser Schicht ist es, die Daten für die höhere Schicht 7 zu interpretieren. Deswegen wird diese Ebene Anpassungsschicht genannt.

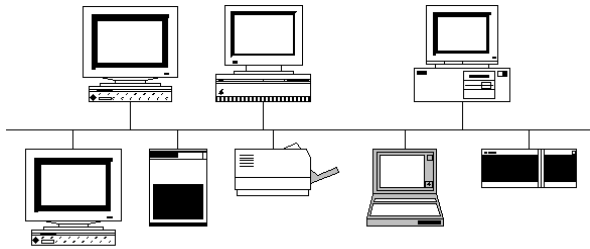
### 7. Anwendungsschicht (Application Layer)

Die oberste Schicht des OSI-Modells definiert die Schnittstelle für die eigentlichen Anwendungen. Sie ist also die Schnittstelle zwischen Programm und Anwender.

## Netzwerktopologien

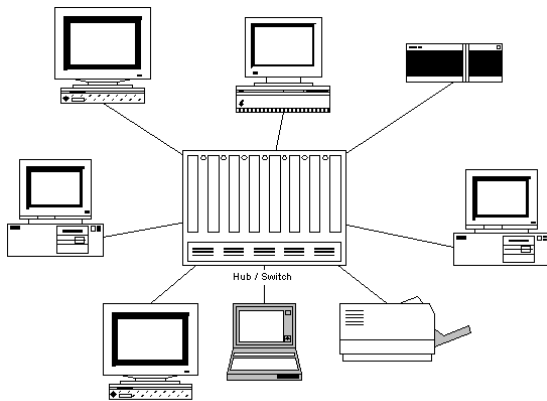
Netzwerke benutzen eine physische Verbindung, um eine Kommunikation aufzubauen. Dabei kann es sich um eine *Kabel-*, eine *Funk-* oder eine *optische* Verbindung handeln. Die heute meist genutzten Verbindungsmedien sind kabelbasierte Netzwerke. Die Art einer Verbindung bezeichnet man als Topologie. In lokalen Netzwerken unterscheidet kennt man drei Grundformen der Netzwerkstruktur:

### Bustopologie



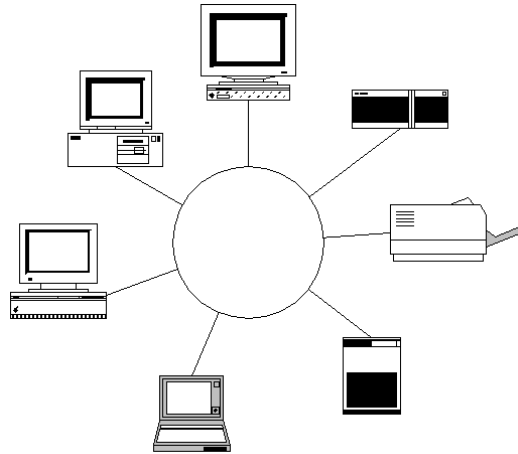
In dieser Topologie werden alle Rechner an ein gemeinsames Übertragungsmedium angeschlossen. Vorteile dieser Topologie sind geringe Erstellungskosten für ein Netzwerk sowie seine leichte Erweiterbarkeit. Der Netzwerkbus wird in der Regel an den beiden Enden durch Endwiderstände abgeschlossen.

### Sterntopologie



Hier werden alle Rechner an zentrale und aktive Netzwerkkomponenten angeschlossen. Diese aktiven Komponenten können *Hubs*, *Switches* oder *Server* sein. Da jede Station ein eigenes Kabel zum zentralen Verbindungsgerät benötigt, sind die Kosten hier etwas höher; das wird aber mehr als ausgeglichen durch die größere Sicherheit: ein defektes Kabel zieht hier keinen kompletten Ausfall des Netzwerkes nach sich.

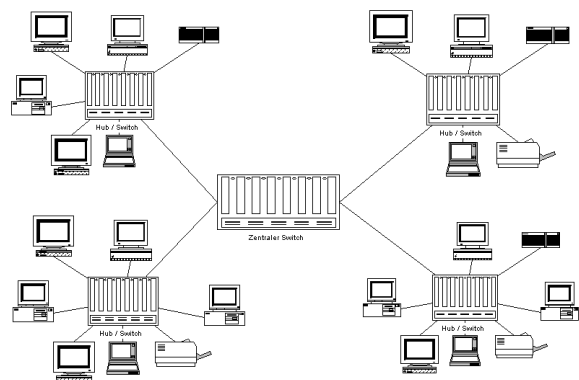
### Ringtopologie



Wie schon in der Bustopologie gibt es auch bei der Ringtopologie ein gemeinsames Übertragungsmedium. In Gegensatz zum Bus liegt hier ein in sich geschlossener Ring vor, an welchem alle Stationen angeschlossen werden. Dabei übergibt eine Station die Daten an ihren Nachbarn, die diese Informationen weiterreicht bis zum Empfänger. In der Praxis wird diese Topologie in neuen Netzwerken jedoch kaum noch eingesetzt.

### Baumtopologie

Die Baumtopologie, die auch „Topologie der strukturierten Verkabelung“ genannt wird, ist eigentlich keine eigene Struktur, sondern vielmehr ein Zusammenschluß mehrerer Sterne, bei welchen die aktiven Komponenten gemäß den Spezifikationen der strukturierten Verkabelung verbunden werden.



## Vernetzungsarten und Geschwindigkeiten

Natürlich gehört zu einem Netzwerk mehr als nur die Topologie. In vielen Topologien können mehrere Geschwindigkeiten eingesetzt werden. Ich gehe hier lediglich auf die verbreitetsten ein.

### 10 - Base - 2

Die Vernetzung nach 10-Base-2 ist eine ältere Vernetzungstechnik. Sie basiert auf der Bustopologie und verwendet ein Koaxialkabel vom Typ RG58 (auch Cheapernet oder Thin-Ethernet genannt). Die maximale Buslänge darf bei dieser Technik maximal 185 Meter betragen, kann aber durch *Repeater* um weitere Segmente verlängert werden. Ein Netzwerk darf hier höchstens 5 Segmente enthalten, von denen lediglich 3 mit Stationen bestückt sein dürfen. Die übrigen zwei Segmente können nur als reine *Verbindungssegmente* eingesetzt werden, z.B. um größere Entfernungen zu überbrücken.

Das Konzept hat seine Vorteile in der Einfachheit und in der günstigen Realisierung. Die Nachteile beruhen vor allem in den allgemeinen Nachteilen eines Bussystemes und den Nachteilen von Datenübertragungen über Koaxialkabel, die leider eine gute Basis für elektrische Störungen bieten.

Die Übertragung erfolgt mit 10Mbit in der Sekunde.

### 10 - Base - T

Ein 10-Base-T-Netzwerk folgt der *Stern-* oder *Baumtopologie*. Zum Einsatz kommen dabei mindestens Kabel der Kategorie 3, meist werden aber heute Kabel der Kategorie 5 verwendet. Die Datenübertragung erfolgt über zwei verdrehte Kupferadern. Dank dieser Technik ist die Datenübertragung deutlich weniger fehleranfällig für elektrische Feldstörungen.

Die Übertragungsgeschwindigkeit im Netzwerk beträgt auch hier 10Mbit in der Sekunde.

### 100 - Base - T

Die Vernetzung nach 100-Base-T folgt so gut wie komplett den Richtlinien der 10-Base-T-Vernetzung. Die Unterschiede liegen vor allem in den Kabeln, die hier der Kategorie 5 entsprechen müssen. Die Datenübertragungsgeschwindigkeit beträgt 100 Mbit in der Sekunde.

## Weitere Typen

Neben der Vernetzung mittels Kupferkabel gibt es natürlich auch Vernetzungstechniken auf der Basis von Glasfasern oder Funktechniken. Darauf gehe ich hier jedoch nicht ein, da derartige Vernetzungstechniken sehr teuer sind und in kleineren und mittleren Netzwerken nur selten zum Einsatz kommen.

## Protokolle im Netzwerk

Soll eine erfolgreiche Kommunikation mit allen angeschlossenen Stationen im Netzwerk realisiert werden, müssen die angeschlossenen Rechner die gleiche Sprache sprechen und bestimmte Kommunikationsregeln einhalten – was im Grunde ja bei jeder Kommunikationsart der Fall ist. Die Regeln und Sprache im Netzwerk nennt man *Kommunikationsprotokoll*. Es gibt unterschiedliche Protokolle für Netzwerke, die jeweils Sprachumfänge von unterschiedlichem Umfang haben – und nicht jedes Protokoll ist für jeden Einsatzzwecke sinnvoll.

Gab es zu Anfang des Vernetzezeitalters noch sehr viele Hersteller-abhängige Protokolle, die eine Interoperabilität verhinderten, so haben sich mittlerweile einige Standards herausgebildet. Im PC-Netzwerk findet man derzeit hauptsächlich drei verschiedene Protokolle: *Netbios*, *IPX* und *TCP/IP*.

### NetBIOS

Das NetBIOS – die Abkürzung steht für *Network Basic Input/Output System* – war die ursprüngliche Schnittstelle für DOS-Computer zur Kommunikation in Netzwerken. Vorgestellt wurde dieses Protokoll 1981 von IBM. Auch heute noch wird es hauptsächlich in der Windows-Welt benutzt. Sein Einsatzbereich sind vorwiegend kleinere Netzwerke. Einer der größten Mankos dieses Protokolles ist, dass es über keinerlei Routingfunktionen verfügt und somit für große oder verteilte Netzwerke nicht verwendbar ist.

### IPX

Das IPX-Protokoll findet man vorherrschend in Novell-Umgebungen. Novell wählte dieses Protokoll in den achtziger Jahren als Standardprotokoll für die Fileserver der Firma und verschaffte ihm damit einen großen Markt. IPX

kann als Kommunikationsplattform auf allen gängigen Medien aufsetzen und besitzt auch Routingfunktionen, so daß es auch in verteilten Netzwerken zum Einsatz kommt.

### TCP/IP

TCP/IP ist das derzeit wohl bekannteste Protokoll. Es ist sehr ausgereift und verfügt über ausgezeichnete Routing-Funktionen, wodurch es sich hervorragend für den Einsatz in grossen und verteilten Netzwerken eignet. Da kann es denn auch nicht wundern, daß dieses Protokoll zur Kommunikation im Internet eingesetzt wird. Durch die rasche Verbreitung des Internet hat sich auch das TCP/IP-Protokoll überall eingenistet. Früher sah man TCP/IP vor allem in der Unixwelt – heute ist es aber für so gut wie alle Betriebssysteme und Hardwarearchitekturen verfügbar, ja es hat sich fast zum Standardprotokoll der PC-Netzwerktechnik entwickelt. Das Unix-derivat Linux hat einen großen Einsatzbereich im Internet, weswegen es großen Gebrauch von TCP/IP macht – also werde ich auf dieses Protokoll später genauer eingehen.

## Unterschiedliche Arten von Netzwerken

Es gibt zwei Hauptformen von Netzwerken, die man nach ihrer Funktionalität unterscheidet: *Peer-to-Peer*-Netzwerke und *Client-Server*-Netzwerke.

### Peer-to-Peer-Netzwerke

Bei *Peer-to-Peer*-Netzwerken sind alle Computer im Netzwerk gleichberechtigt. Es gibt hier keinen Server – alle Rechner können auf Ressourcen aller anderen Stationen zugreifen und sie verwenden.

Dieser Netzwerktyp eignet sich jedoch nur für kleinere Netzwerke. Dabei kann es sich typisch um kleinere MS-Windowsumgebungen handeln.

### Client-Server-Netzwerke

Innerhalb einer *Client-Server*-Struktur gibt es klar verteilte Aufgaben. Hier übernimmt ein oder mehrere Rechner die Funktion eines Servers. Ein solcher Server hat vielfach ein anderes Betriebssystem als die angeschlossenen Arbeitsstationen.

Typische Serverbetriebssysteme sind Unix, Novell NetWare oder MS Windows NT Server. Die Arbeitsstationen werden dagegen meist unter

Windows 95/98, Windows NT Workstation oder Apple's MacOS betrieben.

Aufgrund der klaren Aufgaben in solchen Netzwerken ist dieser Netzwerktyp für große Systeme bestens geeignet, da man hier eine zentrale Administration der wichtigen Systeme hat und allen angeschlossenen Systemen identische Funktionen bereitstellen kann.

### Mischumgebungen

In komplexeren und in sehr großen Umgebungen findet man oftmals auch Mischumgebungen aus *Peer-to-Peer*- und *Client-Server*-Netzwerken. Hier gibt es Ressourcen, die zentral bereitgestellt werden, aber auch zumindest Teilnetzwerke, deren Arbeitsstationen auch Zugriffe auf andere Arbeitsstationen haben.

Solche Umgebungen findet man vor allem in heterogenen Betriebssystemumgebungen. So können Windows-Arbeitsstationen bestimmte Ressourcen für andere Benutzer freigeben, und Arbeitsstationen auf Unixbasis können außerdem noch – je nach Konfiguration – verschiedene Serverdienste anbieten.

Bei Licht betrachtet findet man nur ganz selten eine 'reine' Umgebung. De facto hat man es fast immer mit gemischtem Umgebungen zu tun, bei denen die grundsätzliche Struktur in Richtung *Client-Server* geht.

## Grundlagen des TCP/IP-Protokolls

Linux und das TCP/IP-Protokoll sind eng verbunden. Zwar ist das im Internet verwendete Netzwerkprotokoll älter, aber da der Vertrieb von Linux immer über das Internet erfolgte, lag seit Beginn der Linux-Entwicklung ein Schwerpunkt auf den TCP/IP-Netzwerkfunktionen.

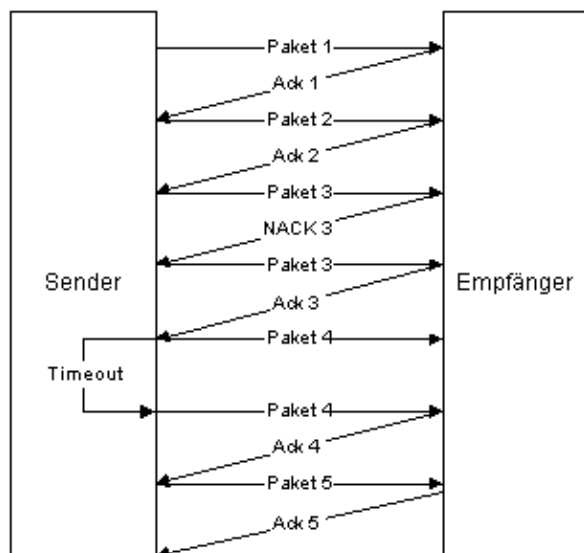
Will man ein Netzwerk auf TCP/IP-Basis einrichten und verwalten, sind zunächst einige Informationen über dieses Protokoll erforderlich.

### Funktion und Aufbau

TCP/IP ist ein Paket-orientiertes Protokoll – was bedeutet, dass die Informationen in kleineren Paketen verschickt werden. Jedes dieser Pakete hat einen eindeutigen Aufbau: es besteht aus einem Datenteil sowie aus einem Paketkopf, dem sogenannten *Header*, der Informationen für die Kommunikation sowie Informationen über den Aufbau des jeweiligen Pakets enthält.

Sind die Informationen größer als der Datenteil eines Paketes, werden sie auf mehrere Pakete aufgeteilt. Auf der Grundlage von Paketnummern im Header setzt der Empfänger die Pakete zusammen. Während der Übertragung werden empfangene Pakete bestätigt. Kommt ein Paket defekt an, erfolgt eine negative Bestätigung an den Sender, der das defekte Paket erneut sendet. Erfolgt nach einer festgelegten Zeit keine Bestätigung oder *Timeout* für ein Paket, wird es erneut gesendet.

Eine TCP/IP-Kommunikation sieht etwa so aus:



Zu beachten ist, dass TCP/IP sich aus zwei Teilprotokollen zusammensetzt: dem *Transmission Control Protocol* TCP und dem *Internet Protocol* IP. Während IP für die Adressierung und die Datenübertragung zuständig ist, kümmert sich TCP um die korrekte Übertragung, indem es Prüfsummenberechnungen durchführt.

### Adressierung im TCP/IP-Netzwerk

Soll ein Protokoll alle Computer im Netzwerk ansprechen, benötigt es ein eindeutiges Adressierungsschema. Grundlage dieses Systems sind die berühmten *IP-Adressen*. Jeder Rechner erhält eine eindeutige Adresse, über welche er im Netzwerk erreichbar ist. Diese Adresse besteht aus einer Kombination von vier Bytes, die jeweils durch einen Punkt getrennt werden.

Um festzulegen, ob Sender und Empfänger im gleichen Netzwerk liegen, gibt es eine weitere Konfigurationsangabe im TCP/IP-Protokoll: die sogenannte *Netzmaske* – eine Bitmaske, die den Host- vom Netzwerkanteil trennt.

Mittels der eigenen IP-Adresse und der eigenen Netzmaske sowie der IP-Adresse des Partners kann der Sender feststellen, ob es sich um eine lokale Übertragung handelt: die IP-Adresse des Senders und die des Empfängers werden in einer logischen UND-Operation mit der eigenen Netzmaske verknüpft. Sind die Ergebnisse dieser zwei Operationen gleich, liegt der Empfänger im eigenen Netzwerk; sind sie ungleich, befindet er sich in einem entfernten Netzwerk, und das Paket wird an einen Gateway gesendet.

Das folgende Beispiel mag diesen Prozeß verdeutlichen:

- Senderadresse: **192.168.1.10**  
(Binär: **1100 000. 1010 1000. 0000 0001. 0000 1010**)
- Netzmaske: **255.255.255.0**  
(Binär: **1111 1111. 1111 1111. 1111 1111. 0000 0000**)
- Empfängeradresse: **192.168.2.2**  
(Binär: **1100 0000. 1010 1000. 0000 0010. 0000 0010**)

Soll ermittelt werden, ob der Empfänger im lokalen Netzwerk ist, werden folgende Verknüpfungen durchgeführt:

Verknüpfung 1:

```

11000000 10101000 00000001 00001010
&& 11111111 11111111 11111111 00000000
11000000 10101000 00000001 00000000

```

Verknüpfung 2:

```

11000000 10101000 00000010 00000010
&& 11111111 11111111 11111111 00000000
11000000 10101000 00000010 00000000

```

Vergleicht man die beiden Ergebnisse, wird klar, dass sie nicht identisch sind. Der Empfänger liegt nicht im eigenen Netzwerk – also muß das aktuelle Paket an einen Router gesendet werden, der es an den korrekten Empfänger weiterleitet.

Genaugenommen wird durch die erste Verknüpfung die eigene Netzwerkadresse ermittelt. Diese Adresse repräsentiert jeweils ein komplettes Netzwerk – also besitzen alle Rechner innerhalb eines Netzwerks die gleiche Netzwerkadresse. Diese ist immer die erste IP-Adresse eines Adressblocks. Sie nimmt also eine Sonderstellung ein und darf nicht an einen Computer vergeben werden. Eine weitere besondere Rolle hat die letzte Adresse eines IP-Adressblocks. Sie ist die sogenannte Broadcast-Adresse. Auf diese Adresse hören alle Rechner in einem Netzwerk – Sendungen an diese Adresse werden also von allem Rechnern im jeweiligen Netzwerk entgegengenommen. Auch die Broadcastadresse darf nicht an einen Rechner vergeben werden.

Soll ein Computer mit Systemen außerhalb des eigenen Netzwerkes kommunizieren, benötigt er die Adresse eines Routers, welcher nicht-lokale Pakete an das jeweilige Zielnetzwerk weiterleitet. Diese Adresse nennt man *Gateway*.

Logischerweise muß das Gateway innerhalb des eigenen Netzwerkes liegen. Ein besonderes

Gateway ist ein *Router*, welcher alle Pakete bekommt, die nicht lokal zustellbar sind oder für deren Zielnetzwerk es kein explizites Gateway gibt. Diesen Router nennt man *Standard-* oder *Default-Gateway*.

Damit wären alle Eckdaten für eine TCP/IP-basierte Datenübertragung benannt: eigene IP Adresse, Netzmaske, Broadcast und Default Gateway. Mit diesen Daten ist es möglich, einen Rechner in ein TCP/IP-Netzwerk einzubinden.

Da die komplette Internetkommunikation auf dem TCP/IP Protokoll basiert, ist es logisch, dass große Teile des möglichen Adressraumes für Computer im Internet reserviert sind. Damit es nicht zu Doppelbelegungen von Adressen im Internet kommt, werden diese von zentralen Organisationen in den USA, Europa und Asien vergeben. Für Netzwerke ohne direkte Internetanbindung (also hinter Firewalls oder in lokalen Netzwerken) sind spezielle Adressräume reserviert, welche im Internet nicht benutzt werden. Diese Adressen sind im Internetstandard oder *RFC – Request for Comment – 1918* definiert und lauten:

- 192.168.0.0 bis 192.168.255.255
- 172.16.0.0 bis 172.31.255.255
- 10.0.0.0 bis 10.255.255.255

Für lokale Netzwerke sollte auf jeden Fall auf diese Adressräume zurückgegriffen werden, auch im eigenen Interesse – z.B. um Probleme bei einem evtl. späteren Internetanschluß von Anfang an aus dem Weg zu räumen.

### **Ports - Die Services im Netzwerk**

Neben den IP Adressen gibt es innerhalb der IP Kommunikation noch eine weitere Angabe: die sogenannten Ports, die mit Kanälen vergleichbar sind. Für jede Datenübertragung wird ein solcher Kanal benutzt, der sich aus einem Quellport und einem Zielport zusammensetzt. Desweiteren werden solche Ports benutzt, um bestimmte Services oder Dienstleitungen anzusprechen. So können etwa Serverprozesse an einzelne Ports 'gebunden' werden. Dies bedeutet, dass Anfragen an einen bestimmten Kanal immer an ein zugewiesenes Programm weitergeleitet werden, z.B. an eine Webserversoftware. Eine TCP / IP-Verbindung besteht also grundsätzlich aus einer Ursprungsadresse, einem Ursprungsport, einer Zieladresse und einem Zielport. Mittels dieser Angaben ist eine Übertragung zwischen zwei Netzwerkstationen genau identifizier- und zuordbar.

Einige dieser Ports sind standardmäßig bestimmten Anwendungen vorbehalten. Diese sind in entsprechenden RFC's nachlesbar.

## Einrichten eines IP-Netzwerks unter Linux

Die erfolgreiche Navigation durch ein Netzwerk setzt voraus, dass zunächst einmal ein solches eingerichtet wurde. Wie man das macht, hängt von der jeweiligen Distribution ab. Die meisten Distributionen liefern eigene Administrations-Tools mit. So verwendet SuSE den eigenen *YaST*, RedHat hat zur Netzwerkkonfiguration das grafische Tool *netcfg*, während Debian *netconfig* mitliefert. Natürlich ist das interne Vorgehen bei all diesen Tools relativ identisch – wie man die Sache auch von Hand erledigen kann. Allerdings werden die TCP/IP-Daten meist schon bei der Installation abgefragt.

Sehen wir uns nun die Einrichtung eines TCP/IP-Netzwerkes am Beispiel der RedHat-Distribution an, unter Verwendung des freien Tools *linuxconf*. Ich habe mich zu diesem Tool aus zwei Gründen entschlossen. Zum einen legt RedHat es standardmäßig seiner Distribution bei und empfiehlt seine Verwendung, zum anderen tun das immer weitere Distributionen, so dass *linuxconf* auf dem Weg zu einem distributionsübergreifenden Administrationswerkzeug ist.

Die Einrichtung verläuft eigentlich recht einfach. Nach dem Start des Programms wählt man den Menüpunkt [CONFIG | NETWORKING | CLIENT TASKS | BASIC HOST INFORMATION](#)

```

pan: Linuxconf 1.16 (subrev 3-2)
- Config
- Networking
- Client tasks
  Basic host information
  Name server specification (DNS)
- Routing and gateways
  Defaults
  other routes to networks
  other routes to hosts
  routes to alternate local nets
  the routed daemon
  Host name search path
  Network Information System (NIS)
  IPX interface setup
  PPP/SLIP/PLIP
  Quit  Ctrl-S  Help

```

Im Menüpunkt [BASIC HOST INFORMATION](#) werden die TCP/IP-Eckdaten, sowie die Daten zur Netzwerkkarte eingegeben. Damit ist der Grundstein zum Anschluß an ein IP Netzwerk gelegt:

```

This host basic configuration
You are allowed to control the parameters
which are specific to this host and related
to its main connection to the local network

Host name      pan
Config mode    [X] Enabled
               [O] Manual [ ] Dhcp [ ] Bootp
Primary name + domain pan.wusel.de
Aliases (opt) pan
IP address     212.14.64.9
Netmask (opt) 255.255.255.240
Net device     eth0
Kernel module nezak-pci
I/O port (opt)
Irq (opt)
  Accept  Cancel  Help

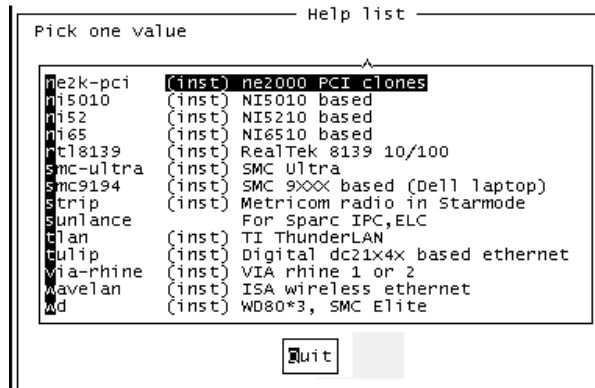
```

Der Menüpunkt [HOST NAME](#) fragt nach dem Rechnernamen. Dieser kann frei gewählt werden. Über den Menüpunkt [CONFIG MODE](#) wird festgelegt, ob die TCP/IP-Daten manuell – und somit fix – hinterlegt oder mittels *DHCP* oder *BOOTP* beim Systemstart zugewiesen werden. Hierbei ist folgendes zu beachten: soll der Computer Serverdienste anbieten, ist es wichtig, dass er immer unter der selben IP-Adresse zu erreichen ist. Hier sollte man eine manuelle und fixe Konfiguration wählen, oder dafür sorgen, daß der DHCP-Server dem eigenen Rechner immer die gleiche Adresse zuweist. Übernimmt der Rechner dagegen reine Workstation-Aufgaben und bietet keinerlei Serverdienste im Netzwerk an, kann eine IP-Konfiguration via DHCP durchaus sinnvoll sein – natürlich muß es im Netzwerk dann einen DHCP-Server geben, der entsprechende Anfragen beantwortet. Sinnvoll ist dies vor allem in großen Netzwerken, da dann die IP-Daten der Arbeitsstationen vom Administrator zentral festgelegt werden können.

Der Menüpunkt [PRIMARY NAME + DOMAIN](#) fragt nach dem sogenannten *Full qualified Domain Name*, dem *FQDN*. Dies bedeutet Rechnername und Domainname. In meinem Beispiel arbeite ich am Rechner **pan** in der Domain **wusel.de**. Im Punkt [ALIASES \(OPT\)](#) kann man zusätzliche Aliasnamen angeben. Als nächstes werden die IP-Adresse und die Netzwerkmaske des Rechners erfragt.

Im weiteren Verlauf wird das Netzwerkgerät („Net device“) erfragt. Gemeint ist hier die Netzwerkkarte, welche mit der zugeordneten IP-Adresse arbeiten soll. Da meist nur eine Karte in

einem Rechner ist, ist dies im allgemeinen das Gerät *eth0* (Ethernetkarte Null). Anschließend muß der Treiber der Netzwerkkarte benannt werden. Im aktuellen Rechner wird eine günstige PCI-Netzwerkkarte mit RTL-Chipsatz benutzt, was dem Modul *ne2k-pci* entspricht. In *linuxconf* sind die Treiber aus einem Auswahlmenü auszuwählen, in welchem die Kartennamen neben den Treibernamen aufgelistet sind:



Des weiteren müssen für ISA-Karten noch einige Daten zur verwendeten Hardware angegeben werden, wie IRQ und Basisadresse. Bei PCI-Karten werden diese Informationen vom Treiber selbst bei der Karte erfragt.

Sind alle Informationen eingegeben, werden die Daten mit einem Druck auf **ACCEPT** übernommen. Beim Verlassen von *linuxconf* werden die Konfigurationsdaten aktiviert.

Voraussetzung für eine erfolgreiche Konfiguration ist jedoch, dass die benötigten Kartentreiber als Module vorliegen oder im Kernel enthalten sind. Des weiteren muß die Option **TCP/IP NETWORKING** im Kernel enthalten sein. Dies ist jedoch bei den generischen Kernels, die von den Distributoren geliefert werden, meist der Fall.

Mittels der getätigten Angaben ist eine Kommunikation in einem lokalen Netzwerk möglich. Soll der Rechner auch mit Systemen außerhalb seines eigenen Netzwerkes in Verbindung stehen, muss ein „Standardgateway“ angegeben werden – normalerweise ein Router, der über Informationen zu weiteren Netzwerken verfügt und Pakete entsprechend weiterleitet.

Diese Angabe kann ebenfalls mittels *linuxconf* gesetzt werden, und zwar im Menüpunkt **ROUTING AND GATEWAYS | DEFAULTS**. Hier wird einfach die IP-Adresse des Gateways angegeben.

Natürlich ist *linuxconf* lediglich ein komfortables Frontend, das die nötigen Daten in den relevanten Konfigurationsdateien des Linuxsystems speichert und ggf. aktiviert. Unter *RedHat Linux* werden die IP-Konfigurationsdaten in folgenden Dateien gespeichert:

- **/etc/sysconfig/network**: Hier sind Daten wie Hostname und Gateway hinterlegt.
- Die eigentlichen Daten der TCP/IP-Konfiguration finden sich im Verzeichnis **/etc/sysconfig/network-scripts**. Dort gibt es für jede Netzwerkkarte eine Datei **ifcfg-eth-X** – das X steht für die Nummer der Netzwerkkarte. Für die gezeigte Konfiguration in *linuxconf* sieht diese Datei **ifcfg-eth0** aus wie folgt:

```
DEVICE="eth0"
USERCTL=no
ONBOOT="yes"
BOOTPROTO="none"
BROADCAST=212.14.64.15
NETWORK=212.14.64.0
NETMASK="255.255.255.240"
IPADDR="212.14.64.9"
```

Natürlich kann der Systemverwalter diese Dateien auch von Hand editieren. Bei der Initialisierung des Netzwerkes greift das System direkt auf sie zu und startet über die darin enthaltenen Informationen das Netzwerk.

## Test des IP-Netzwerks

Ist das Netzwerk erfolgreich eingerichtet, sollte seine Funktionalität getestet werden. Für diesen Zweck stellt die TCP/IP-Protokoll-Suite einige leistungsfähige Befehle zur Verfügung.

### PING - Testen der Verbindung und des TCP/IP Protokoll Stacks

*Ping* ist ein Befehl, der mehrere Pakete an den Kommunikationspartner sendet und die Zeit bis zur Ankunft des jeweiligen Antwortpakets mißt. Es handelt sich um ein Kommandozeilen-orientiertes Programm, das als Parameter lediglich die Ziel-Adresse benötigt. Optional kann man dem Befehl noch die Größe des zu sendenden Pakets mitteilen. Die Syntax lautet wie folgt, wobei der gesamte Befehl in einer Zeile steht:

```
Ping [-s Paketgröße]
<Ziel-IP-Adresse>
```

Als Information liefert Ping die Größe des gesendeten Paketes, die IP-Adresse des Zielrechners sowie die Laufzeit des Paketes. Ist der *Ping*-Befehl erfolgreich, funktioniert zumindest die grundsätzliche Kommunikation. Will man die Funktion des eigenen TCP/IP-Protokollstacks prüfen, sollte man zunächst einmal die eigene IP-Adresse *pingen*. Gelingt dies, kann man davon ausgehen, daß zumindest der eigene Stack richtig arbeitet. Schlägt der *Ping* fehl, sendet man im nächsten Schritt einen *Ping* auf die *Loopback-Adresse*. Diese ist bei jedem Rechner unter dem IP-Protokoll die Adresse **127.0.0.1** – unter dieser Adresse erreicht man grundsätzlich den eigenen Rechner. Die Adresse wird beim Einrichten des Protokollstacks automatisch konfiguriert. Gelingt dieser *Ping*, liegt vermutlich ein Fehler in der Konfiguration vor, nicht aber in der Installation des Protokollstacks. Ein Beispiel:

```
[as@hades as]$ Ping 212.14.64.2
PING 212.14.64.2 (212.14.64.2): 56 data bytes
64 bytes from 212.14.64.2: icmp_seq=0 ttl=64 time=1.6 ms
64 bytes from 212.14.64.2: icmp_seq=1 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=2 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=3 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=4 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=5 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=6 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=7 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=8 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=9 ttl=64 time=1.5 ms
64 bytes from 212.14.64.2: icmp_seq=10 ttl=64 time=1.5 ms

--- 212.14.64.2 Ping statistics ---
11 packets transmitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 1.5/1.5/1.6 ms
[as@hades as]$
```

## Fehlerquellen und Diagnosewerkzeuge

Natürlich kann es auch beim Einrichten eines Netzwerkes zu Problemen kommen. Auch hier stellt Linux Hilfswerkzeuge zur Verfügung.

Die Prüfung der eigenen IP-Adresse sowie der Loopback-Adresse **127.0.0.1** mittels *Ping* haben wir uns bereits angesehen. Klappte das, kann man davon ausgehen, dass das TCP/IP-Modul arbeitet. Die nächste Fehlerdiagnose richtet sich auf die Konfiguration der IP Daten.

Ergibt eine Überprüfung dieser Daten keinen Hinweis auf einen Fehler, könnte es ein Problem im Bereich der Netzwerkkarte geben. Also sollte man zunächst prüfen, ob der Netzwerkkartentreiber richtig geladen ist. Bei aktuellen Kernelversionen werden die Treiber im allgemeinen als Modul hinzugeladen – siehe auch Seite 13. Mit dem Befehl **lsmod**, der alle geladenen Module anzeigt, prüft man, ob das Modul geladen ist:

```
[as@hades as]$ lsmod
Module                Pages      Used by
autofs                 2           1 (autoclean)
nfs                    12          5 (autoclean)
ne2k-pci               1           2 (autoclean)
8390                   2      [ne2k-pci]    0 (autoclean)
[as@hades as]$
```

In dieser Ausgabe ist das Modul *ne2k-pci* zu sehen, das Modul für die entsprechende Netzwerkkarte. Es wäre möglich, dass das Modul mit falschen Einstellungen z.B. der IRQ oder der Basisadresse arbeitet, die ja bei älteren ISA-Bus-Karten richtig konfiguriert sein müssen – im Gegensatz zu modernen PCI-Karten, wo man diese Parameter nicht mehr einstellen muß.

Die Konfiguration der Kernelmodule erfolgt in der Datei `/etc/conf.modules`.

Sind auch diese Angaben richtig, sollte die Interfacekonfiguration kontrolliert werden – also inwieweit die IP-Konfiguration der Netzwerkkarte richtig zugeordnet ist und die IP-Daten korrekt an das Interface gebunden wurden.

Die Interfacekonfiguration kann mittels des Befehles **ifconfig** überprüft werden. Soll die Konfiguration aller Interfaces angezeigt werden, übergibt man dem Befehl beim Aufruf die Option **-a** (wie *all*).

```
as@hades as]$ ifconfig -a
lo                Link encap:Local Loopback
                  inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
                  UP BROADCAST LOOPBACK RUNNING MTU:3584 Metric:1
                  RX packets:43173 errors:0 dropped:0 overruns:0
                  TX packets:43173 errors:0 dropped:0 overruns:0

eth0              Link encap:Ethernet HWaddr 00:00:CB:23:61:48
                  inet addr:212.14.64.2 Bcast:212.14.64.15 Mask:255.255.255.240
                  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                  RX packets:514001 errors:0 dropped:0 overruns:0
                  TX packets:306174 errors:0 dropped:0 overruns:0
                  Interrupt:11 Base address:0xe800
```

Aus der Ausgabe geht hervor, dass der Rechner über zwei logische Netzwerkgeräte verfügt.

Zunächst steht hier das **lo**-Device, bei dem es sich um das bereits erwähnte *Loopback-Device* handelt. Wie man sieht, ist diesem Gerät die IP-Adresse **127.0.0.1** zugeordnet, was ja auch korrekt ist.

Des Weiteren wird angezeigt, wieviele Pakete empfangen – *RX* – und gesendet – *TX* – wurden. Darüber hinaus werden die bei der Kommunikation aufgetretenen Fehler angezeigt – *Errors*, *Drops* und *Overruns*. Im Idealfall treten natürlich keine Fehler auf.

Als zweites Interface sieht man die Schnittstelle **eth0**, also die erste Ethernetkarte. Hier wird ebenfalls die komplette IP-Konfiguration angezeigt. Da es sich um ein physisches Interface und nicht um ein logisches handelt, sieht man hier neben statistischen Daten für empfangene, gesendete und fehlerhafte Pakete Hardwareeinstellungen wie IRQ und Basisadresse.

Sind die Hardwareparameter richtig eingestellt, sollte auch die angezeigte *MAC-Adresse* angezeigt werden, die Hardwareadresse der Netzwerkarte. Ist diese Adresse auf einen unrealistischen Wert eingestellt, etwa

**00:00:00:00:00:00** oder auch **ff:ff:ff:ff:ff:fe**, kann man davon ausgehen, dass falsche Daten für IRQ oder Speicheradresse vorliegen.

Werden auf dem Ethernetgerät die richtigen Hardwareeinstellungen angezeigt, obwohl viele Fehlermeldungen wie z.B. *TX/RX-Errors* auftreten, kann man auf ein defektes Übertragungsmedium schließen. Da die Netzwerkkarte richtig erkannt und konfiguriert wurde, sollten also die verwendeten Kabel oder aktiven Netzwerkkomponenten wie *Hubs* oder *Switches* überprüft werden.

Endlich kommt es auch vor, dass sowohl der eigene Rechner als auch die Rechner im lokalen Netzwerk erreichbar sind, z.B. mittels *Ping*, nicht aber Rechner in entfernten Netzwerken. In diesem Fall kann man meist auf Routingprobleme schließen. Dies könnte im einfachsten Fall eine falsche oder fehlende Einstellung für das Standardgateway sein. Hier prüft man zunächst die eigenen Routing-Einstellungen, was am einfachsten über den Befehl *netstat* geht. Sollen die Routing-Einstellungen in einer numerischen Ausgabe erscheinen, übergibt man die Parameter **-rn** (*r* für *Routing*, *-n* für *numerisch*):

```
[as@hades as]$ netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
212.14.64.0 0.0.0.0 255.255.255.240 U 1500 0 0 eth0
127.0.0.0 0.0.0.0 255.0.0.0 U 3584 0 0 lo
0.0.0.0 212.14.64.1 0.0.0.0 UG 1500 0 0 eth0
```

Die Routingtabelle sollte eine Route für das eigene, also das lokale Netzwerk haben sowie eine Route für das Loopback-Netzwerk – grundsätzlich **127.0.0.0**. Die Route für das lokale Netzwerk sollte dabei auf die eigene Ethernetkarte zeigen – hier auf **eth0** –, die Route für das Loopbacknetzwerk auf das **lo**-Interface. Bei beiden Netzwerken wird kein Eintrag für ein Gateway benötigt.

Sollen Rechner außerhalb des eigenen Netzwerks erreicht werden, muß eine entsprechende Route existieren. Normalerweise wird hierzu ein Standardgateway eingetragen, das alle Pakete erhält, die nicht örtlich zustellbar sind. Eine

*Default-Route* wird grundsätzlich mit dem Ziel **0.0.0.0** gekennzeichnet. Das entsprechende Gatewaysystem muß innerhalb des lokalen Netzwerks liegen und als Gateway für diese Route eingetragen sein – in unserem Beispiel also das System mit der IP-Adresse **212.14.64.1**

Da das Routersystem im eigenen Netzwerk sein muß, zeigt diese Route natürlich auf das Interface **eth0**, da der Router über diese Schnittstelle erreichbar ist.

Fehlt dieser Eintrag, sollte man das Standardgateway in der Netzwerkkonfiguration eintragen. Ist dieser Eintrag korrekt, und es sind dennoch keine

entfernten Systeme erreichbar, ist es möglich, daß der Fehler in einem entfernten Netzwerk liegt. Will man den Weg eines IP-Paketes zu seinem Ziel verfolgen, bedient man sich des

Befehls *traceroute*, der alle Router eines IP Paketes bis hin zu seinem Ziel anzeigt. Als Parameter übergibt man lediglich die Zieladresse:

```
[as@hades as]$ traceroute www.linuxinfo.de
traceroute to www.linuxinfo.de (192.67.198.4), 30 hops max, 40 byte packets
 1  karlsfeld.de.wusel.NET (212.14.64.1)  2.574 ms  2.174 ms  2.190 ms
 2  xave.Muenchen.ISAR.net (212.14.65.20) 33.014 ms 28.612 ms 32.808 ms
 3  fe-0-0-0.Muenchen.ISAR.net (212.14.65.30) 30.185 ms 27.027 ms 25.688 ms
 4  muenchen2.core.xlink.net (212.14.65.130) 26.907 ms 27.263 ms 26.816 ms
 5  stuttgart.core.xlink.net (194.122.234.10) 31.474 ms 31.808 ms 31.100 ms
 6  karlsruhe.core.xlink.net (194.122.227.129) 34.442 ms 35.152 ms 111.827 ms
 7  karlsruhe.core.xlink.net (194.122.243.17) 33.121 ms 33.096 ms 33.488 ms
 8  www.linuxinfo.de (192.67.198.4) 33.971 ms 34.601 ms 34.476 ms
```

Unser Beispiel gibt Auskunft über den Weg vom lokalen Rechner zu einem Webserver namens [www.linuxinfo.de](http://www.linuxinfo.de). Hier werden unterwegs sieben weitere Stationen berührt. Ist der Weg nicht erreichbar, weil eines der Systeme temporär nicht verfügbar ist, wird das aus der Ausgabe des *traceroute*-Befehles ersichtlich.

Es wurden hier nur einige Möglichkeiten zur Fehlerdiagnose gezeigt. Natürlich können auch individuelle Fehler vorliegen – in diesem Fall ist der detektivische Spürsinn des Lesers gefragt...

## Die Nutzung von Netzwerken

Soviel zur Einrichtung eines Netzwerks. Das folgende Kapitel widmet sich seiner Nutzung aus der Anwendersicht. Wir beginnen mit der Clientseite und sehen uns zunächst die Nutzung der üblichsten Netzwerkservices an.

### Terminaldienste

*Terminaldienste* werden alle Services genannt, die es ermöglichen, eine Terminalsession auf einem entfernten Computer zu öffnen und so seine Programme und Ressourcen zu nutzen.

Terminaldienste sind textorientiert. Linux bietet mehrere Arten solcher Terminalverbindungen an.

### Telnet

Telnet ist eine sehr alte Anwendung der Netzwerktechnik. Es bietet eine komplette Terminalnutzung des entfernten Rechners über das Netzwerk – lediglich Ein- und Ausgabe werden auf das eigene System umgeleitet. Die Nutzung von Ressourcen und der Arbeitsprozess erfolgen auf dem entfernten Computer. Das ermöglicht zum einen die Fernadministration von Systemen; zum anderen erleichtert es die Verteilung der Ressourcen, die ja auf dem Rechner verfügbar sind, wo ein Programm gestartet wird. Der lokale Rechner benötigt lediglich die Ressourcen für die Ein- und Ausgabe. Werden Anwendungen auf zentralen Servern gestartet, können die Arbeitsplatzrechner geringer ausgestattet sein, ohne dass das eine Leistungseinbuße bedeutete.

Telnet ist ein Kommandozeilenprogramm. Als Parameter erhält es die IP-Adresse oder den Namen des entfernten Rechners, mit dem Verbindung aufgenommen wird. Ein Beispiel:

```
telnet www.linuxinfo.de
```

```
[as@hades as]$ rsh -l as 192.168.1.1
Password:
Last login: Thu Dec  9 20:56:44 from www.linuxinfo.de
[as@trillian as]$
```

Wie man sieht, wird nach der Befehlsausführung nach dem *Password* des gewünschten Benutzers gefragt, das interaktiv eingegeben werden muss – worauf man dann in einer normalen Shell auf dem fernen Computer arbeitet.

Will man nun aber einzelne Befehle automatisiert auf anderen Rechnern ausführen lassen, ist eine

### r tools – rsh und rlogin

Weitere Möglichkeiten der Terminalemulation bieten zwei Programme der sogenannten *r-tools*, einer Sammlung von Programmen für die kommandozeilenorientierte Nutzung eines Computers über das Netzwerk. Das **r** steht hier für *remote*, also entfernt.

Das Programm **rsh** bietet eine *Remote Shell*, was bedeutet, dass man mit seiner Hilfe über das Netzwerk eine *Shell* – also eine *Terminalsession* – auf einem entfernten Computer ausführen kann. Das besondere am **rsh**-Befehl ist die Möglichkeit, Variablen zu übergeben oder Befehle auf anderen Systemen auszuführen. Dadurch eignet sich dieser Befehl für Automatisierungen über das Netzwerk: er veranlasst einen entfernten Computer dazu, einzelne Befehle abzuarbeiten. Werden keine Parameter mit dem Befehl übermittelt, aktiviert er eine interaktive Shell auf dem fernen Computer, wodurch man dann ganz normal interaktiv arbeiten kann. Wie in Terminal-sitzungen via *Telnet* werden alle Ressourcen des ferngesteuerten Rechner außer der Ein- und Ausgabe genutzt.

Die Syntax dieses Befehles lautet:

```
rsh [Optionen] <Rechnername oder IP
Adresse> [Befehl]
```

Standardgemäß versucht sich das Programm mit der *Userkennung*, also dem Benutzernamen anzumelden, unter welcher man den Befehl auf dem lokalen System aufgerufen hat. Will man sich auf dem fernen System unter einer anderen Kennung anmelden, erfordert das den Parameter **-l** **<Username>**. Ein Beispiel:

interaktive Passwordeingabe recht hinderlich. Das umgeht man, indem man jedem Rechner mitteilt, welcher Benutzer sich von welchem Rechner aus ohne Password-Eingabe mittels der **r Tools** anmelden darf.

Diese Informationen werden im eigenen Homeverzeichnis in der Datei **.RHOSTS** hinterlegt.

Damit kann man bestimmten Benutzerkennungen von festgelegten Rechnern aus den unproblematischen Anmeldevorgang ermöglichen. Die Datei enthält die Benutzerkennung des entfernten Benutzers sowie den Rechnernamen bzw. die IP-Adresse des Rechners, wo sich diese Kennung befinden muss. Ein Beispiel: will man sich von Rechner **A** mit der eigenen Kennung **as** auf Rechner **B** anmelden können, muss die **.RHOSTS-**

Datei des Users **as** auf Rechner **B** folgendermaßen aussehen:

```
[as@trillian as]$ cat .rhosts
212.14.67.190 as
[as@trillian as]$
```

Wie man einen Befehl – hier **ls -l** – auf dem entfernten Rechner ausführt, zeigt Beispiel 2:

```
[as@hades as]$ rsh -l as 212.14.67.189 ls -la
total 16
drwx----- 3 as as 1024 Dec 10 19:19 .
drwxr-xr-x 4 root root 1024 Sep 4 20:13 ..
-rw-r--r-- 1 as as 1422 Sep 4 20:13 .xdefaults
-rw----- 1 as as 2851 Dec 10 19:21 .bash_history
-rw-r--r-- 1 as as 24 Sep 4 20:13 .bash_logout
-rw-r--r-- 1 as as 230 Sep 4 20:13 .bash_profile
-rw-r--r-- 1 as as 124 Sep 4 20:13 .bashrc
-rw----- 1 as as 17 Dec 10 19:19 .rhosts
-rw-rw-r-- 1 as as 3505 Sep 4 20:13 .screenrc
drwxr-xr-x 2 as as 1024 Nov 30 19:39 .ssh
[as@hades as]$
```

Wie man sieht, ist man vor und nach der Befehlsausführung auf dem lokalen Rechner angemeldet. Lediglich der übergebene Befehl wird auf dem entfernten Rechner ausgeführt.

Der **rlogin**-Befehl arbeitet analog zum **rsh**-Befehl, falls letzterer ohne Parameter aufgerufen wird. Er liefert die Kommandoshell auf dem fernen Computer.

### s Tools – ssh und slogin

Ein Nachteil der **r-Tools** – den sie übrigens mit Telnet teilen – ist, dass die Übertragung des eigenen Passwortes und der Daten über das Netzwerk im Klartext erfolgt – wodurch die Kommunikation zwischen den Partnersystemen mittels entsprechender Werkzeuge 'abhörbar' ist.

Die Programme **ssh** und **slogin** gehören zu den **s-Tools**. Diese sind an die **r-Tools** angelehnt und analog zu benutzen, haben aber einen großen Vorteil: sie arbeiten mit verschlüsselter Kommunikation, wobei auch die Übertragung des Passwortes verschlüsselt ist.

Auch die Auswertung der **.RHOSTS**-Datei kann vom Systemadministrator erlaubt werden. Um sie vor unbefugten Zugriffen zu schützen, sollte aber grundsätzlich nur der Eigentümer Lese- und Schreibzugriff auf sie haben. Manche Systeme verweigern ohnehin die Benutzung dieser Datei, falls andere Rechte auf sie vergeben wurden.

## Die Nutzung von Dateidiensten

Unter Dateidiensten versteht man Services, die sich mit der Dateibearbeitung befassen. Natürlich kann man auch über das Netzwerk Dateidienste nutzen. Voraussetzung ist, dass ein Netzrechner diese Dienste anbietet – und dass man berechtigt ist, diese Möglichkeiten zu nutzen.

### File Transfer Protocol - ftp

Das *File Transfer Protocol* – abgekürzt *FTP* – ist eines der ältesten Protokolle im Internet. Es dient dazu, Dateien über das Netzwerk zu übertragen. Um das Protokoll zu nutzen, bedient man sich eines sogenannten *FTP-Client* – also eines Programms, das über dieses TCP/IP-basierte Protokoll mit einem Server Verbindung aufnimmt und die Dateiübertragung regelt.

Einer der ältesten FTP-Clients ist das Kommandozeilen-orientierte Programm **ftp**, das man auf fast jeder UNIX-Maschine findet. Dank seiner Textorientiertheit lässt es sich problemlos von Rechnern ausführen, auf die z.B. mittels **ssh** über das Netz zugegriffen wird. Natürlich gibt es heute auch FTP-Programme mit einem grafischen Userinterface oder *GUI*.

Das Programm **ftp** wird aufgerufen, indem man dem Programmnamen **ftp** als Parameter den Namen oder die Adresse des Zielrechners übergibt. Auf dem Netzrechner meldet man sich mittels Benutzername und Kennwort an, worauf man in einer Art *FTP-Shell* arbeitet – siehe das nachfolgende Beispiel. Die Shell stellt bestimmte Anweisungen zur Verfügung, von denen die wichtigsten in der Tabelle auf der nächsten Seite beschrieben werden.

```
[as@hades as]$ ftp ftp.wusel.de
Connected to ftp.wusel.de.
220 ftp.wusel.de FTP server (Version wu-2.4.2-academ[BETA-18](1) Mon Aug 3
19:1
7:20 EDT 1998) ready.
Name (ftp.wusel.de:as): schmidt
331 Password required for schmidt.
Password:
230 User schmidt logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

<i>Anweisung</i>	<i>Parameter</i>	<i>Erklärung</i>
<b>ls</b>		Anzeigen des aktuellen Verzeichnisinhaltes
<b>cd</b>	Verzeichnisname	Wechseln in das angegebene Verzeichnis des fernen Rechners. Wird als Name .. (zwei Punkte) angegeben, so wird in das nächsthöhere Verzeichnis gewechselt. Die Adressierung ist analog dem Shellbefehl <b>cd</b> .
<b>bin</b>		Wechseln in den Binärmodus. Dies ist ratsam, falls binäre Dateien, meist Programmdateien, übertragen werden sollen
<b>get</b>	Dateiname	Übertragung der angegebenen Datei vom entfernten Rechner auf den eigenen
<b>put</b>	Dateiname	Übertragung des angegebenen Datei vom eigenen Rechner auf den entfernten. Dazu wird eine entsprechende Schreibberechtigung auf dem entfernten Rechner benötigt.
<b>lcd</b>	Verzeichnisname	Wechseln in das angegebene Verzeichnis auf dem eigenen Rechner. Alle Notationen des <b>cd</b> -Befehles der Shellumgebung sind hier nutzbar
<b>pwd</b>		Zeigt das aktuelle Arbeitsverzeichnis auf dem entfernten Rechner an.
<b>mget</b>	Dateinamen	Überträgt eine oder mehrere Dateien vom entfernten Rechner an den lokalen Rechner. Dabei können mehrere Dateinamen angegeben werden. Diese müssen durch ein Leerzeichen getrennt werden. Des weiteren ist die Nutzung von <i>Jokerzeichen</i> wie dem Sternchen zum Ersetzen des Dateinamensrestes möglich.
<b>mput</b>	Dateinamen	Überträgt eine oder mehrere Dateien vom lokalen Rechner auf den entfernten. Die Parametrierung ist analog dem <b>mget</b> -Befehl möglich.
<b>help</b>	Befehlsname	Zeigt eine Hilfe zu dem übergebenen <b>ftp</b> -Befehl an. Wird kein Parameter übergeben, werden alle bekannten <b>ftp</b> -Anweisungen angezeigt.
<b>bye</b>		Abmelden vom fernen Server

Das folgende Beispiel zeigt die Übertragung der Datei `TEST-1` von einem entfernten auf den lokalen Rechner. Nach der `ftp`-Sitzung ist sie unter gleichem Namen auf dem lokalen Rechner verfügbar, und zwar im aktuellen Arbeitsverzeichnis. Da es sich um die Übertragung einer Programmdatei handelt, wurde vor dem eigentlichen Download, den der Befehl `mget` aktiviert, mit der Anweisung `bin` in den binären Übertragungsmodus gewechselt.

Wird das bei einer Programmdatei unterlassen, kommt sie unter Umständen nicht korrekt an, da die ASCII-Übertragung des `ftp`-Protokolls nicht 8-bit-konform arbeitet – was dazu führt, dass Zeichen aus dem erweiterten Zeichensatz fehlerhaft ankommen und das Programm unbrauchbar wird. Handelt es sich beim Download nicht um eine reine ASCII-Datei, sollte dieser Moduswechsel immer vorgenommen werden – z.B. auch bei komprimierten Dateien.

```
[as@hades as]$ ftp ftp.wusel.de
Connected to ftp.wusel.de.
220 ftp.wusel.de FTP server (Version wu-2.4.2-academ[BETA-18](1) Mon Aug 3
19:1
7:20 EDT 1998) ready.
Name (ftp.wusel.de:as): schmidt
331 Password required for schmidt.
Password:
230 User schmidt logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> promp
Interactive mode off.
ftp> bin
200 Type set to I.
ftp> mget test-1
local: test-1 remote: test-1
200 PORT command successful.
150 Opening BINARY mode data connection for test-1 (386 bytes).
226 Transfer complete.
386 bytes received in 0.0537 secs (7 Kbytes/sec)
ftp> bye
221 Goodbye.
[as@hades as]$
```

Einige FTP-Server bieten einen speziellen Dienst an – das sogenannte *anonyme* (oder *anonymous*) *ftp*. Damit ist es möglich, Dateien eines Servers zu kopieren, obwohl man dort keine eigene Userkennung besitzt. Da man sich beim File Transfer Protocol jedoch anmelden muß, wurden hierfür zwei spezielle Kennungen geschaffen: zum einen der Benutzer *anonymous*, zum anderen das Konto *ftp*. Die meisten Server verlangen nach Eingabe des Benutzernamens ein Passwort. Dieses wird oftmals nicht geprüft – und könnte somit beliebig gewählt werden. Es hat sich jedoch eingebürgert,

dass man seine eigene Emailadresse eingibt. Manche Server prüfen, ob es sich bei dem eingegebenen Passwort um eine gültige Mailadresse handelt. Es ist also nicht etwa nur ein Gebot der Höflichkeit, sich an dieses ungeschriebene Gesetz zu halten – dieses Vorgehen verspricht auch den meisten Erfolg.

Nach einer solchen anonymen Anmeldung hat man einen Lesezugriff auf den angesprochenen FTP Server, normalerweise aber aus Sicherheitsgründen keinen Schreibzugriff.

## rcp - Dateiübertragung mit r-Tools

Das Programm **rcp** ist ein weiterer Bestandteil der auf Seite 19 erwähnten *r Tools*. Dieses *Remote Copy Program* arbeitet ähnlich wie sein lokales Pendant **cp**. Sollen allerdings auf diese Weise Dateien zwischen zwei Systemen übertragen werden, setzt das eine Benutzerkennung auf beiden Rechnern voraus – ein anonymer Datentransfer ist hier nicht möglich. Deswegen wird dieses Programm meist in lokalen Netzwerken genutzt. Ein weiterer Vorteil des Programms ist, dass hier die **.RHOSTS**-Datei auf dem Netzrechner ausgewertet wird, wodurch man eine Passwortabfrage umgehen kann. Also lässt sich dieses Tool für automatisierte Aufgaben verwenden.

Als Parameter werden dem **rcp**-Befehl zum mindesten Quell- und Zielort übergeben. Im Gegensatz zum **cp**-Befehl, der ausschließlich lokal arbeitet, muss der **rcp**-Anweisung auch der Netzrechner mitgeteilt werden. Dies erreicht man durch eine in Netzwerken übliche Notation für Dateipfade, die den Rechnernamen vor dem lokalen Pfad auf dem Zielsystem und von diesem durch einen Doppelpunkt getrennt einsetzt.

Der schematische Aufbau einer solchen Angabe sieht so aus:

**Rechneradresse:Dateipfad**

Rechneradresse kann hier eine IP-Adresse oder ein gültiger DNS-Name sein. Wird keine Rechneradresse angegeben, geht die Anweisung davon aus, dass es sich um einen Dateipfad auf dem lokalen System handelt.

Im folgenden Beispiel wird eine Datei vom eigenen Rechner auf einen entfernten Rechner in `/tmp` kopiert:

```
[as@hades as]$ rcp test-1
192.168.1.1:/tmp
```

Will man diesen Befehl mittels einer anderen Benutzerkennung auf dem entfernten Rechner durchführen, kann diese vor die Rechneradresse gestellt werden. Dabei werden Benutzername und Rechneradresse durch das `@`-Zeichen getrennt:

```
[as@hades as]$ rcp test-1
schmidt@192.168.1.1:/tmp
```

Existiert auf dem fernen System keine **.RHOSTS**-Datei, oder ist die eigene lokale Kennung dort nicht berichtigt, wird man zur Eingabe des korrekten Passwords aufgefordert.

Natürlich können auch hier die Jokerzeichen '\*' und '?' benutzt werden. In diesem Zusammenhang sind zwei weitere Parameter des **rcp** Kommandos interessant:

<b>-p</b>	kopiert die angegebenen Dateien unter Beibehaltung der ursprünglichen Dateirechte
<b>-r</b>	veranlaßt den Befehl, rekursiv zu arbeiten – evtl. vorhandene Unterverzeichnisse werden also mitkopiert. Dies kann natürlich nur dann geschehen, wenn man ein komplettes Verzeichnis als Quelle angibt.

## scp - Dateien sicher über das Netzwerk kopieren

Ein analoges Programm zum **rcp**-Befehl gibt es in den auf Seite 20 erwähnten **S TOOLS**. Es arbeitet im grossen und ganzen wie **rcp**, hat aber wie alle **S TOOLS** den Vorteil, dass sowohl Passwordeingaben als auch die komplette Dateiübertragung verschlüsselt ablaufen. So lassen sich Dateien auch über größere Netzwerke sicher kopieren. Gerade im Internet-Zeitalter ist dies eine sehr sinnvolle Ergänzung.

## nfs – der transparente Zugriff auf das Dateisystem über das Netz

Das *Network File System* **nfs** macht es möglich, Dateisystemteile auf Netzrechnern transparent in das eigene Dateisystem zu integrieren. Dies ist dann sinnvoll, wenn man dauernd mit Dateien arbeitet, die sich auf Rechnern im Netzwerk befinden – will man nur einzelne Dateien oder Verzeichnisse kopieren, ist **rcp** bzw. **scp** vorzuziehen. Hier ist zu beachten, dass sich die Dateien physisch auf dem entfernten Rechner befinden – sie werden also nicht kopiert, sondern komplett ins eigene Dateisystem integriert. Eine Autorisierung findet auf IP-Ebene statt: auf dem Server wird festgelegt, welche IP-Adressen diesen Dienst benutzen dürfen und welche Teile des Dateisystemes über **nfs** zugänglich sind.

Technisch ist das Einbinden recht einfach. Zum einen muß der Kernel dieses Protokoll unterstützen – was die meisten Linuxkernels tun –, zum anderen muß der Server die Verbindung erlauben. Sind diese Voraussetzungen gegeben, kann

man das entfernte Dateisystem einfach mittels des **mount**-Befehles einbinden:

```
mount 192.168.1.1:/exports/home
/home
```

Hier wird das Verzeichnis **/exports/home** des Rechners **192.168.1.1** als lokales Verzeichnis **/home** ins eigene Dateisystem eingebunden. Da dieses Protokoll einen tiefen Eingriff in die Administration des lokalen und des fernen Rechners bedeutet, sollte der Befehl nur vom Benutzer **root** ausgeführt werden dürfen.

Will man einen solchen **mount**-Vorgang automatisieren, so dass die gewünschten Verzeichnisse beim Start des eigenen Rechners unmittelbar eingebunden werden, wird ein entsprechender Eintrag in der Filesystem-Tabelle des Rechners vorgenommen, also der Datei [/ETC/FSTAB](#).

Allerdings hat das **nfs**-Protokoll einige grobe Sicherheitsprobleme – es bietet hervorragende Arbeitsbedingungen für Hacker. Man sollte das System also nur in einem lokalen Netzwerk benutzen, das von einem öffentlichen Netzwerk wie dem Internet durch Schutzmechanismen wie z.B. eine *Firewall* getrennt ist.

### smbmount – der Zugriff auf Windowsfreigaben im Netzwerk

Unter Linux lassen sich auch Dateifreigaben von Windowssystemen transparent in das eigene Dateisystem integrieren. Voraussetzung ist, dass das Programmpaket *Samba* auf dem Linuxrechner installiert ist. Aus Sicherheitsgründen sollte diese Möglichkeit wiederum dem Benutzer **root** vorbehalten sein.

Dem Befehl **smbmount** wird als Parameter der Rechnername der Windowssystemes, der Name der zu mountenden Freigabe und der aktuelle Windows-Benutzer sowie sein Passwort übergeben. Anschließend hat man von Linux aus freien Zugang auf dieses Dateisystem mit allen lokalen Tools und Programmen:

```
smbmount //ntserver/unixexport
/winimport -U ntuser -P ntpwd
```

Dieser Befehl mountet die Freigabe **UNIXEXPORT** des NT-Servers **NTSERVER** – mit Hilfe der Windows-Benutzerkennung **NTUSER** und des entsprechenden Passwords.

Wiederum gilt, dass eine Benutzung dieser Möglichkeiten über offene Netzwerke gefährlich ist, da wie bei **nfs** die komplette Kommunikation unverschlüsselt stattfindet.

## Mail, Netnews und Webdienste

*Elektronische Post*, *Usenet Netnews* und das *World Wide Web* – diese drei sind derzeit wohl die bekanntesten und meistgenutzten Dienste im Internet. Hier folgt nur ein kurzer Überblick – das KnowWare-Heft *Start ins Internet* befasst sich genauer mit diesem Thema.

### Electronic Mail

... dient zur Übermittlung von persönlichen Nachrichten zwischen zwei oder mehreren Teilnehmern. Dieser Dienst erfordert einen Server, der die ausgehende Post verschickt und die ankommende Post aufbewahrt, bis sie abgeholt wird. Einen solchen Server nennt man einen *SMTP-Server* – nach dem *Simple Mail Transfer Protocol*, das Post transferiert, also überträgt. Damit die eingegangenen Nachricht vom Empfänger abgeholt werden können, bedarf es eines weiteren Dienstes, der nach dem hierzu benutzten *Post Office Protocol* als POP-Server bezeichnet wird. Der *Mailclient* öffnet eine Verbindung zu diesem Server und überträgt die eingegangenen Nachrichten auf die lokale Arbeitsstation.

*Mailclients*, die man auch *Mail User Agent* nennt, sind eine weit verbreitete Gattung. Unter Linux werden meist der *Netscape Communicator* benutzt, der X-Windows erfordert, oder das Programm *Kmail*, das die KDE voraussetzt. Die meistbenutzten *textbasierten* Programme sind **MUTT** oder **ELM**. Diese Programme haben keine grafische Oberfläche, was sie schnell macht – dafür muss man aber auf eine Inline-Anzeige beigefügter Grafiken verzichten.

### Usenet Netnews

... sind die öffentlichen Foren im Internet. Was hier veröffentlicht wird, ist jedem zugänglich, und jeder kann es beantworten. Man benötigt nur den Zugriff auf einen sogenannten *Newsserver*, der die Nachrichten der öffentlichen Foren speichert und die Informationen zum Lesen mittels des *Net News Transport Protocol*, des **NNTP**, anbietet. Außerdem gibt ein solcher Newsserver die empfangenen Nachrichten an andere Newsserver

weiter. Diese Server arbeiten in Gruppen zusammen und gleichen sich gegenseitig ab – eine neue Nachricht wird also sehr schnell weltweit verteilt.

Als Programm zum Lesen und Schreiben eigener *Newsartikel* oder *Postings* kommt ein *Newsclient* zum Einsatz. Auch hier gibt es viele verschiedene Exemplare: mit grafischer Oberfläche z.B. den *Netscape Communicator* oder *Knews*, im textorientierten Bereich **tin**, **slrn** oder **nn**.

## Webdienste

Das WWW ist wohl der meistgenutzte Dienst im Internet. Es ermöglicht die multimediale Darstellung von Inhalten. Übrigens ist das WWW ein Musterbeispiel für verteilte Systeme: einheitlich dargestellte Inhalte können hier von unterschiedlichen Servern kommen. Der Benutzer merkt dies normalerweise gar nicht, da die Herkunft der einzelnen Elemente nicht angegeben wird.

Zur Darstellung der Inhalte benötigt man einen *Webbrowser*. Der bekannteste ist der *Netscape Navigator*, der zur *Netscape Communicator*-Suite gehört.

## Zeit- und Datumsdienste

Mit Zeit- und Datumsdiensten werden Systemdatum und –zeit synchronisiert – und zwar nicht nur auf einem, sondern auf mehreren Rechnern gleichzeitig. Dies lässt sich auf verschiedene Weisen erreichen, es setzt aber in jedem Fall die Existenz eines Zeitserver voraus, der die aktuelle Zeit über das Netzwerk bereit stellt.

### rdate

Das Programm `rdate` ist ein einfaches Tool, das die Zeit bei einem beliebigen Rechner abfragt. Außerdem bietet es die Möglichkeit, die erhaltene Zeit als eigenes Systemdatum zu setzen.

Das erfordert aber Systemverwalterprivilegien – mit anderen Worten muss das Programm also als **root** ausgeführt werden.

Als Parameter wird die IP-Adresse oder der Name des Servers angegeben, von welchem die Zeit bezogen wird.

**rdate** verfügt über zwei Optionen:

Option	Bedeutung
<b>-p</b>	zeigt das Datum des entfernten Rechners an
<b>-s</b>	setzt das empfangene Datum / Zeit als eigenes Systemdatum und -zeit

Da dieses Programm aus der Kommandozeile aufgerufen werden muss, eignet es sich vor allem für die Synchronisation von Rechnern, die nur temporär mit dem Internet verbunden sind, z.B. über eine Wählverbindung.

### xntp

Das **xntp/ntp**-Programm ist ein *Daemon* zur Zeitsynchronisierung. Ein *Daemon* ist ein Programm, das konstant im Hintergrund aktiv ist. **xntp/ntp** eignet sich also in erster Linie für Systeme mit ständiger Netzwerkanbindung.

Konfiguriert wird das Programm über die Datei `/ETC/NTP.CONF`

Die Datei für die Konfiguration als Client ist sehr einfach herzustellen– sie sieht so aus:

```
peer ntp.isar.net
peer ntp2.isar.net

driftfile /var/run/ntp.drift
pidfile /var/run/xntpd.pid
```

Soviel zur Nutzung von Netzwerkdiensten. Sehen wir uns nun an, wie man diese Dienste betreibt.

## Die Betreuung von Netzwerkdiensten

Serverdienste laufen im Hintergrund und werden über das Netzwerk mit den Clients in Verbindung gesetzt. Mittels definierter Protokolle übertragen sie Informationen zwischen Server und Client.

Je nach Dienst wird ein entsprechendes Server- oder Anbieterprogramm installiert und konfiguriert.

Zur Kommunikation benötigen diese Programme feste *Ports* des IP-Protokolls, über die sie angesprochen werden. Serverprozesse können sich entweder unmittelbar an 'ihren' Port binden oder mit Hilfe eines *Startprogramms*, das mit dem jeweiligen Port verbunden ist und bei eingehenden Verbindungen das Serverprogramm startet.

### inetd

Ein solcher 'Starthelfer' ist **inetd**. Er wird über die Datei `/ETC/INETD.CONF` konfiguriert und ist mit allen Ports verbunden. Wird ein Port angerufen, startet **inetd** das zugeordnete Programm. Da es als übergeordneter Starthelfer genutzt wird, also sozusagen als Serverprozess für viele Server-

programme, wird dieses Programm oftmals auch als Superserver bezeichnet.

Zu beachten ist, dass ein Port jeweils nur an ein Serverprogramm gebunden werden kann. Belegt ein Programm unmittelbar einen Port, steht dieser also nicht mehr für **inetd** zur Verfügung.

Die für **inetd** relevante Datei `/ETC/INETD.CONF` enthält alle Angaben, die benötigt werden, um einen Serverprozess zu starten:

- Serviceport
- Socket-Type
- das zu verwendende Protokoll
- benötigte Flags
- der User, unter welchen der Prozess läuft
- das auszuführende Serverprogramm
- evtl. benötigte Programmoptionen

Eine typische `/ETC/INETD.CONF` sieht so aus (Ausschnitt):

```
#
# inetd.conf      This file describes the services that will be available
#                through the INETD TCP/IP super server.  To re-configure
#                the running INETD process, edit this file, then send the
#                INETD process a SIGHUP signal.
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
daytime      stream      tcp      nowait      root    internal
daytime      dgram      udp      wait        root    internal
ftp          stream      tcp      nowait      root    /usr/sbin/tcpd  in.ftpd -l
-a
telnet       stream      tcp      nowait      root    /usr/sbin/tcpd  in.telnetd
pop-3        stream      tcp      nowait      root    /usr/sbin/tcpd  ipop3d
tftp         dgram      udp      wait        root    /usr/sbin/tcpd  in.tftpd
```

In dieser Datei sind einige Dienste für den Start via **inetd** konfiguriert. Zeilen, die mit dem *Hash*(#)-Zeichen beginnen, sind Kommentare ohne Einfluß auf den Ablauf des Programms. Grundsätzlich können fast alle Netzwerkdienste via **inetd** gestartet werden. Der Vorteil dieses Systemes ist, daß das Programm die Netzwerkkommunikation bearbeitet – sein Nachteil, dass der Aufbau

einer Verbindung etwas länger dauert und dass diese Startart bei intensiver Nutzung eines Dienstes deutlich mehr Ressourcen benötigt. Daher ist es manchmal sinnvoller, einen Serverprozess – z.B. den Webserver – direkt auf einen Port zu binden. Das Problem entsteht, weil für jede Verbindung eines neuen Clients ein neuer Serverprozess zu starten ist.

## Die Betreuung von Terminaldiensten

Ein Terminaldienst oder eine Terminalemulation erlaubt einem Benutzer den Anschluss an den Server von einem beliebigen Netzwerkcomputer aus, um eine *Terminalsession* zu betreiben. Es gibt mehrere Möglichkeiten für Terminaldienste.

### Telnet

Im allgemeinen wird der Telnetserver **telnetd** über **inetd** gestartet, also nicht unmittelbar an einen Port gebunden. Eine besondere Konfiguration des Servers ist nicht nötig. Anpassungen der Arbeitsumgebung können während einer

Terminalsession über *Umgebungsvariablen* eingesetzt werden, die jeweils von der genutzten *Kommandoshell* abhängen – z.B. **bash**. Mit Telnet arbeitet man wie auf der lokalen Textkonsole des Servers.

Allerdings ist zu beachten, dass die gesamte Kommunikation des Telnetprotokolls unverschlüsselt über das Netzwerk gesendet wird – also auch übertragene Benutzerkennungen und Passwords.

Die Konfigurationszeile in der **/ETC/INETD.CONF** sieht im allgemeinen so aus:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

### rsh & rlogin

Die Services **rsh** und **rlogin** ermöglichen ebenfalls die Ausführung von Befehlen oder ganzen Terminalsessions über das Netzwerk. Eine nähere Erklärung dieser Dienste findet sich im Kapitel über die Clients auf Seite 19.

```
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
login stream tcp nowait root /usr/sbin/tcpd in.rlogind
```

Wie **Telnetd** benötigen diese Programme keine weiteren Konfigurationsschritte – und leider läuft auch hier die Kommunikation unverschlüsselt ab.

Auch hier werden die Server über **inetd** gestartet, wofür folgende Konfigurationszeilen benötigt werden:

### ssh

Wie auf Seite 20 erwähnt arbeitet das **ssh**-Protokoll wie **rsh** und **rlogin**, nur verschlüsselt. Der SSH-Server **sshd** wird meist unmittelbar gestartet; das bedeutet, dass er eine eigene Konfi-

gurationsdatei benötigt, die **/ETC/SSHD\_CONFIG**, wo alle Optionen für den SSH-Server gespeichert sind. Die Konfigurationsparameter können der **sshd**-Man-Page entnommen werden. Die wichtigsten Parameter sind:

<i>Parameter</i>	<i>Bedeutung</i>
<b>Port</b> <Nummer>	<b>sshd</b> läuft auf dem hier angegebenen Port. Standardport ist der Post 22
<b>ListenAddress</b> <IP Adresse>	IP-Adresse, auf die der Server hört. Wird <b>0.0.0.0</b> angegeben, verbindet sich der Server mit allen konfigurierten IP-Adressen des Systems
<b>HostKey</b> <Path>	Path zur Datei mit dem Host-Schlüssel
<b>PermitRootLogin</b> yes   no	Werden Root-Logins akzeptiert?
<b>IgnoreRhosts</b> yes   no	Soll die Datei <b>.RHOSTS</b> ignoriert werden?
<b>X11Forwarding</b> yes   no	Soll <b>X11</b> -Weiterleitung ermöglicht werden?
<b>AllowHosts</b>	Angabe über Systeme, die sich über <b>ssh</b> anmelden dürfen
<b>DenyHosts</b>	Angabe über Systeme, die sich <i>nicht</i> über <b>ssh</b> anmelden dürfen

Ein Beispiel für eine typische Konfigurationsdatei folgt hier:

```
Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh_host_key
RandomSeed /etc/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility DAEMON
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
AllowHosts *.linuxinfo.de mypc.wusel.de
DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny yes
```

## Dateidienste

Unter Linux kann man viele verschiedene Dateidienste anbieten. Eine Beschreibung ihrer kompletten Bandbreite würde den Umfang dieses Heftes übersteigen – also begnügen wir uns mit den wichtigsten.

### File Transfer Protocol - FTP

Das **ftp**-Protokoll erlaubt Benutzern aus dem Netzwerk den Zugriff auf Teile des Dateisystems eines Servers. Es ist für die reine Übertragung von Dateien optimiert, eignet sich also nicht für das transparente Arbeiten oder eine Eingliederung ins eigene Dateisystem.

Der **ftp**-Dienst erfordert ein entsprechendes Serverprogramm – den sogenannten *FTP-Server*. Es gibt mehrere Exemplare dieser Gattung. Ein zur Zeit sehr beliebter Server ist **proFTPd** – erhältlich unter <http://www.proftpd.org/>. An seinem Beispiel wollen wir uns ansehen, wie die Sache funktioniert.

**proFTPd** kann über **inetd** gestartet werden oder eigenständig laufen. Letztere Variante sollte man ins Auge fassen, wenn der Server stark frequentiert wird – wird er nur ab und zu benutzt, kann man die **inetd**-Methode benutzen.

Konfiguriert wird **proFTPd** über eine eigene Konfigurationsdatei namens **PROFTPD.CONF**, die standardmäßig im Verzeichnis **/USR/LOCAL/ETC** liegt. Hier werden alle Einstellungen für den aktiven Serverprozess vorgenommen.

Nachfolgend erscheint eine Beispielkonfiguration. Allerdings leistet der **proFTPd**-Server viel mehr. Die kompletten Konfigurationsoptionen sind auf der **proFTPd**-Homepage unter <http://www.proftpd.org/> nachzulesen.

```

ServerName "LinuxInfo.DE
  FTP Server"
ServerType                inetd
DefaultServer             on

Port                      21
Umask                     022

MaxInstances              30

User                      nobody
Group                     nobody

<Directory /*>
  AllowOverwrite          on
</Directory>

<Anonymous ~ftp>
  User                    ftp
  Group                   ftp
  UserAlias                anonymous
  ftp

  MaxClients              10

  DisplayLogin             welcome.msg
  DisplayFirstChdir       .message

  <Limit WRITE>
    DenyAll
  </Limit>

</Anonymous>

```

Die gezeigte Konfiguration läßt den Server auf Port 21 laufen und wird über **inetd** gestartet. Die Bedeutung der Parameter ist aus der nachfolgenden Tabelle ersichtlich:

<i>Parameter</i>	<i>Eigenschaft</i>
<b>ServerName</b>	Name des Servers, der beim Login eines Benutzers angezeigt wird.
<b>ServerType</b>	gibt an, ob der Server alleine ( <b>standalone</b> ) oder via <b>inetd</b> (inetd) gestartet wird.
<b>Port</b>	gibt an, auf welchem Port der Server läuft – FTP-Standard ist 21
<b>Umask</b>	gibt an, welche Rechte neu übertragenen Dateien gegeben werden (Serverseitig)
<b>MaxInstances</b>	Anzahl der maximal gleichzeitig möglichen Netzwerkverbindungen
<b>User</b>	Angabe des Benutzers, unter dem der Server läuft
<b>Group</b>	Angabe der Benutzergruppe, unter der der Serverprozess läuft.
<b>&lt;Directory /*&gt; AllowOverwrite on &lt;Directory&gt;</b>	Dateien innerhalb des FTP-Dateisystemes dürfen überschrieben werden.
<b>&lt;Anonymous ~ftp&gt;</b>	Dieses Schlüsselwort leitet die Konfiguration für einen anonymen Zugriff auf den FTP Server ein. Außerdem wird hier festgelegt, für welchen Benutzer der Server gilt und welches das Homeverzeichnis dieses Benutzers ist, das die Daten enthält.
<b>User</b>	Benutzerkennung, unter der der anonyme FTP-Server laufen soll, sowie primäre Loginkennung
<b>Group</b>	Angabe der Gruppe, unter der der Zugriff für diesen Server läuft.
<b>UserAlias</b>	Angabe alternativer Loginnamen – so bedeutet z.B. <b>anonymous ftp</b> , dass das Login <b>anonymous</b> analog zu <b>ftp</b> zu benutzen ist.
<b>MaxClients</b>	Maximale Anzahl gleichzeitiger Netzwerkzugriffe auf den Server/Login
<b>DisplayLogin</b>	Angabe einer Textdatei bzw. eines Pfades zu einer solchen, die jedem neu angemeldeten Benutzer angezeigt wird.
<b>DisplayFirstChdir</b>	Angabe einer Textdatei, die einem Benutzer beim ersten Wechsel in das Verzeichnis angezeigt wird, das diese Datei enthält.
<b>&lt;Limit WRITE&gt;</b>	Schreibzugriff begrenzen
<b>Deny All</b>	Jeglichen Schreibzugriff verbieten
<b>&lt;Limit&gt;</b>	Begrenzung aufgehoben
<b>&lt;/Anonymous&gt;</b>	Ende der <b>Anonymous</b> -Konfiguration

## nfs – das Netzwerkdateisystem

Das Netzwerkdateisystem **nfs** ermöglicht die transparente Nutzung der Dateisystemstruktur eines fernen Rechners über das Netzwerk. Die Konfiguration des Servers und die Freigabe des betreffenden Dateisystembereichs ist denkbar einfach – der **nfs**-Server wird über ein weiteres Hilfsprogramm, den sogenannten Portmapper, gestartet.

Die Clientseite wurde auf Seite 24 beschrieben. Der NFS-Server muss wissen, welche Bereiche

des eigenen Dateisystemes anderen Netzwerkstationen für welche Versionen freizugeben sind. Die Zuordnung wird in der Datei `/ETC/EXPORTS` vorgenommen, mittels der Angabe des oder der Bereiche, zu denen, sowie des oder der Rechnernamen bzw. IP-Adressen, denen der Zugriff erlaubt ist. Die komplette Identifizierung im **nfs** erfolgt auf IP-Basis – es können also keine Benutzernamen oder Passwords verwendet werden. Außerdem erfolgt hier die Definition von Optionen, die für diese Struktur gelten sollen.

```
/tftpboot -network 192.168.10.0 -mask 255.255.255.0
/cdrom -network 192.168.10.0 -mask 255.255.255.0
/exports/home -network 192.168.2.0 -mask 255.255.255.240
/exports/info www.linuxinfo.de web2.linuxinfo.de mail.wusel.de(ro)
```

Diese Datei bewirkt folgendes: alle Rechner aus dem Netzwerk **192.168.10.0** mit der Netzwerkmaske **255.255.255.0** haben Lese- und Schreibzugriff auf den Verzeichnisbaum `/TFTPBOOT`. Dasselbe gilt für die Verzeichnisstruktur unterhalb des Verzeichnisses `/CDROM`. Auf die Struktur `/EXPORTS/HOME` dürfen nur die Rechner aus dem Netzwerk **192.168.2.0** mit der Netzwerkmaske **255.255.255.240** zugreifen, auf die Struktur `/EXPORTS/INFO` nur [WWW.LINUXINFO.DE](http://WWW.LINUXINFO.DE), [WEBS.LINUXINFO.DE](http://WEBS.LINUXINFO.DE) und [MAIL.WUSEL.DE](http://MAIL.WUSEL.DE). Allerdings hat der Rechner [MAIL.WUSEL.DE](http://MAIL.WUSEL.DE) nur Lesezugriff, was angegeben wird durch **ro = readonly**.

Nach einem erfolgreichen Neustart des **mountd** und des **nfsd** können diese Dateien via **nfs** über das Netzwerk gemounted werden, vorausgesetzt, die Anfrage kommt von einer zugelassenen IP-Adresse.

## Samba – Dateidienste für Windowsclients

Natürlich gibt es unter Linux auch die Möglichkeit, Dateisystembereiche für Windowsmaschinen bereitzustellen – mit dem Programm *Samba*, das eine Linux-Dateisystemstruktur als Windowsfreigabe exportiert. *Samba* läuft als eigener Service, der unmittelbar an die entsprechenden TCP/IP-Ports gebunden ist. Der Samba-Server wird in der Datei `/ETC/SMB.CONF` konfiguriert. Da diese Konfigurationsdatei recht komplex ist und *Samba* deutlich mehr kann als nur Dateidienste für Windowsnetze bereitzustellen, erhält dieses Programm sein eigenes Kapitel auf Seite 49.

## Webdienste

Das World Wide Web (WWW) hat sich in letzter Zeit zum Dienst der Dienste im Internet entwickelt. Viele Anwender kennen, abgesehen von Mail und Web, gar keine anderen Dienste.

Nachfolgend werden Einrichtung und Betrieb eines eigenen Webservers beschrieben. Die derzeit wohl bekannteste Serversoftware für WWW-Dienste ist der **APACHE**-Webserver, der unter <http://www.apache.org/> erhältlich ist. Außer seiner weiten Verbreitung hat dieses Programm den großen Vorteil, *Open-Source*-Software zu sein, was bedeutet, dass seine Quelltexte für jedermann einsichtig sind.

Auch ihre Leistungsfähigkeit spricht für diese Software. **APACHE** verfolgt ein modulares Prinzip, das Programm vermag sich also an den geplanten Einsatzzweck anzupassen. Es gibt weiterhin zahlreiche Zusatzmodule, die die Leistungsfähigkeit des Programms extrem steigern. Eine Übersicht der verfügbaren Module findet man im World Wide Web unter der Adresse <http://modules.apache.org/>. Diese Module können entweder unmittelbar in das Programm eingebunden oder dynamisch während der Laufzeit hinzugeladen werden.

Der **APACHE**-Server läuft im allgemeinen als eigener Server und bindet sich selbst an den konfigurierten Port. Die zentrale Konfigurationsdatei des Programms ist sehr umfangreich und komplex. Wir beschränken uns nachfolgend auf die wesentlichen Konfigurationsoptionen. Auf die mögliche Konfiguration als Webproxy gehen wir hier nicht ein.

Die Konfigurationsdatei, die **HTTPD.CONF** heißt, sollte laut der Apache-Source-Distribution im Verzeichnis `/USR/LOCAL/APACHE/CONF/` liegen. Da die Software vielfach mit der Linuxdistribution ausgeliefert wird, liegt sie möglicherweise in einem anderen Verzeichnis. Ein Blick in die Dokumentation der jeweiligen Distribution hilft hier weiter.

In dieser Konfigurationsdatei werden alle zur Laufzeit benötigten Einstellungen getroffen. Nachfolgend werden die wichtigsten Optionen anhand einer Beispielkonfiguration beschrieben.

```
##
## httpd.conf -- Apache HTTP Server Basiskonfiguration (basierend auf
## der Apacheversion 1.3.11
##
## Achim.Schmidt@Wusel.DE
##
## Es handelt sich hier lediglich um eine Basiskonfiguration fuer den
## Betrieb eines einfachen Webserver.
## Weitergehende Dokumentationen sind unter
## der URL http://www.apache.org/ zu finden.

### Section 1: Globale Einstellungen
## Die Anweisungen in diesem Bereich betreffen alle Operationen des
## Apache.

## ServerType kann inetd oder standalone sein. Bei inetd wird der apache
## ueber diesem gestartet und es muss ein entsprechender Eintrag
## in der /etc/inetd.conf vorhanden sein. Performanter und gaengiger
## ist der eigenstaendige Serverbetrieb (standalone)
ServerType standalone

## ServerRoot: Unterhalb dieses Verzeichnisses sind die Dateien
## des Servers (also die Software selbst) sowie seine Konfigurations
## Dateien zu finden. Am Ende darf keine / stehen !
ServerRoot "/usr/local/apache"

## PidFile: Pfad zur PID - Datei (dort wird die Prozessnummer gespeichert)
PidFile /usr/local/apache/logs/httpd.pid

## Timeout: Zeit in Sekunden zu einer Zeitueberschreitung
Timeout 300

## MaxKeepAliveRequests: Die max. Anzahl an gleichzeitigen Anfragen.
## Wird dieser Wert auf 0 gesetzt, ist die Anzahl unbegrenzt.
MaxKeepAliveRequests 100

## KeepAliveTimeout: Anzahl von Sekunden, welche auf dieser Verbindung
## auf den naechsten Request vom gleichen Client gewartet wird.
KeepAliveTimeout 15

## Angabe der mindest und maximalzahl an 'Spare' Serverprozessen. Um genug
## Reserven zu haben, sollten immer mehr Prozesse als Anfragen laufen.
```

```
MinSpareServers 5
MaxSpareServers 10

## Anzahl der Initial - Serverprozesse (beim starten des Webservers)
StartServers 5

## Anzahl der maximal gleichzeitig moeglichen Verbindungen
MaxClients 150

## Anzahl der maximalen Kinderprozesse pro Serverprozess
## Einen unbegrenzten Wert erreicht man mit dem Wert 0
MaxRequestsPerChild 0

## Listen: Diese Angabe kann den Server auf einen bestimmten Port
## oder eine bestimmte IP-Adresse mit Portangabe binden.
## Beispiel:
## Listen 3000
## Listen 12.34.56.78:80

## BindAddress: Mittels dieser Angabe kann der Server auf eine
## bestimmte IP-Adresse binden, oder auf alle konfigurierten
## IP Adressen ( * )
BindAddress *

### Section 2: Hauptserverkonfiguration
## Alle Einstellungen in dieser Gruppe betreffen den Hauptserver

## Port: Der Port, auf welchen ein standalone Server hoert.
Port 80

## Angabe des Benutzers und der Gruppe unter welcher die Serverprozesse
## laufen sollen.
User nobody
Group nobody

## ServerAdmin: Angabe der Emailadresse des Serveradministrators
ServerAdmin webmaster@deine_domain.de

## DocumentRoot: Dies ist das Basisverzeichnis fuer die Dokumente, welche
## der Webserver verarbeiten soll.
DocumentRoot "/usr/local/apache/htdocs"
```

```
## Jedes Verzeichnis, auf welches der Server Zugriff hat, kann mit
## speziellen Zugriffsrechten versehen werden.

<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

## Hier sollten die Rechte fuer das Dokumentenverzeichnis gesetzt werden
<Directory "/usr/local/apache/htdocs">

## Moegliche Optionen sind: None, All oder beliebige Kombinationen aus
## Indexes, FollowSymLinks, ExecCGT oder Multiviews.

    Options All

## Diese Angabe kontrolliert die Moeglichkeiten der .htaccess Datei,
## mit welcher man verschiedene Einzelheiten auch zur Laufzeit des
## Servers aendern kann.
## Moegliche Werte sind: All oder Kombinationen aus Options, FileInfo,
## AuthConfig und Limit

    AllowOverride All

## Kontrolliert den Zugriff
    Order allow,deny
    Allow from all
</Directory>

## UserDir: Webhome der lokalen Benutzer. Dies liegt immer im
## Homeverzeichnis jedes lokalen Users. Der Inhalt ist ueber
## die URL ~benutzer vom Web her erreichbar.
UserDir public_html

## DirectoryIndex: Name der Dateien, welche als Startdokumente
## genutzt werden sollen.
DirectoryIndex index.html index.htm index.cgi index.php index.php3

## AccessFileName: Der Name der Kontrolldatei, welche bei jedem
## Verzeichniszugriff ausgewertet wird.
AccessFileName .htaccess
```

```
## Die folgenden Zeilen schuetzen die .htaccess Datei vor unberechtigtem
## Zugriff
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

## TypeConfig beschreibt wo die mime.types Datei liegt
TypesConfig /usr/local/apache/conf/mime.types

## DefaultType gibt den standard MIME Type an. Normalerweise
## ist dies text/plain
DefaultType text/plain

## HostnameLookups: In den Logdateien werden die Rechnernamen (on) oder
## die Adressen (off) mitprotokolliert.
HostnameLookups On

## ErrorLog: Pfad zur Fehlerlogdatei
ErrorLog /usr/local/apache/logs/error_log

## LogLevel: gibt die Anzahl der protokollierten Informationen an.
## Moegliche Werte sind: debug, info, notice, warn, error, crit, alert und
## emerg
LogLevel warn

## Die folgenden Anweisungen definieren einige Aliases fuer den
## Aufbau von Logzeilen
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""
combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

## Pfad und Typ (vorher definierter Alias) der Access_logdatei
CustomLog /usr/local/apache/logs/access_log combined

## Server soll bei Fehlern oder Hinweisen seine Signatur anzeigen
## Moegliche Werte sind: On, Off, EMail
ServerSignature On
```

```
## Aliases: Hier koennen mehrere Aliases definiert werden. Das Format
## ist:  Alias FakeName Echter_pfad
Alias /icons/ "/usr/local/apache/icons/"

<Directory "/usr/local/apache/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

## ScriptAlias: Diese Angabe gibt an, welches Verzeichnis
## ausfuehrbare Dateien enthaelt, auf welche vom Web her
## zugegriffen werden kann.
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
## Hier werden dann die Rechte fuer dieses Verzeichnis gesetzt
<Directory "/usr/local/apache/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

## Diese AddIcon* Anweisungen sagen dem Server, welches Icon wann zu
## zeigen ist.
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
```

```
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

DefaultIcon /icons/unknown.gif

## AddType erlaubt das Hinzufuegen von eigenen MIME Types.
AddType application/x-tar .tgz

## AddHandler erlaubt das zuordnen von Dateiendungen zu
## sog. Handlern also einem bestimmten Dateityp. Hier werden
## Alle Dateien, welche auf .cgi enden, als CGI Programm
## gekennzeichnet.
AddHandler cgi-script .cgi
```

Zur Steuerung des Servers bringt die Software ein Kontrollprogramm namens [APACHECTL](#) mit, das meist in `/USR/LOCAL/APACHE/BIN` liegt. Es wird mit dem Parameter **start** aufgerufen. Außerdem gibt es noch die Parameter **restart** zum Neueinlesen der Konfigurationsdatei und **stop** zum Beenden des Servers.

Das Kontrollprogramm kennt weitere Parameter für Fehlerdiagnosezwecke: die Parameter **status**, **fullstatus** und **graceful**.

## Web - Proxy, DNS-Server, Mailserver

### Web-Proxydienste

Der Betrieb eines *Proxyservers* kann interessant sein: zum einen kann er dazu dienen, ein lokales Netzwerk vom Internet abzukoppeln, zum anderen spart er Mengen an zu übertragenden Daten, da er Webinhalte zwischenspeichern kann.

Ein Proxyserver nimmt Anfragen eines Client entgegen und sucht die angefragten Seiten in seinem eigenen Zwischenspeicher oder *Cache*. Hat er die gewünschten Informationen nicht selbst verfügbar, ruft er sie aus dem Internet ab, speichert sie und sendet eine Kopie an den anfragenden Rechner. Wird die selbe Seite ein weiteres Mal angefordert, kann er die Informationen dann unmittelbar selbst liefern. Neben den Bandbreiten-Einsparungen bietet das auch einen zusätzlichen Geschwindigkeitsvorteil – ein lokaler Rechner antwortet schneller als ein entferntes System im Internet.

Einer der derzeit unter Linux üblichsten Proxies ist das Programm **squid**. Dabei handelt es sich wieder um freie Software, die, falls sie nicht zum Lieferumfang der benutzten Linuxdistribution gehört, aus dem Internet bezogen werden kann.

Konfiguriert wird das Programm über eine zentrale Konfigurationsdatei namens **SQUID.CONF**. Hier können vielfältige Einstellungen getroffen werden. So ist voll konfigurierbar, wer den Proxy benutzen darf und ob eine Nachprüfung aufgrund der zugreifenden IP-Adresse oder mittels Benutzername und Passwort erfolgen soll. Natürlich sind auch bestimmte Webinhalte sperrbar, so dass man an zentraler Stelle den Zugriff auf Systeme im Internet regeln kann.

Die Konfigurationsdatei ist sehr umfangreich, aber auch sehr gut kommentiert, so dass hier keine ausführliche Behandlung erforderlich ist. Zu beachten ist jedoch, dass der Server so konfiguriert wird, dass nur Rechner aus dem eigenen *Local Area Network* oder *LAN* den Proxy benutzen dürfen, fremde dagegen abgewiesen werden.

### DNS-Server

DNS Server spielen im Internet wie in größeren lokalen Netzwerken eine wichtige Rolle. Ihnen obliegt die Verwaltung des Namensraumes des Netzwerks. Das erfordert zunächst einmal eine

Erklärung der Funktionsweise des DNS. Nachfolgend sehen wir uns den Aufbau eines eigenen DNS-Servers an.

### Wie DNS funktioniert

Die Adressierung in einem TCP/IP-basierten Netzwerk läuft über IP-Adressen. Da wir als Menschen uns diese aus vier Bytes bestehenden Zahlen nur schwer merken können, wurde ein Namenssystem erschaffen: das *Domain Name System* oder *DNS*.

Das DNS gibt die IP Adressen in der Form von Namen wieder und stellt also eine Schnittstelle Mensch-Maschine dar, die einem Benutzer den einfachen und komfortablen Zugriff auf andere Systeme ermöglicht. Da der Namensraum im Internet sehr groß ist, arbeitet DNS mit einer systematischen Strukturierung jedes Rechnernamens in verschiedene Namensteile. Eine volle IP-Adresse besteht aus einem *Rechnernamen*, einer *Subdomain* sowie einer *Top Level Domain*. In größeren Netzwerken können zwischen Rechnernamen und Sub- oder *2<sup>nd</sup> Level-Domain* zusätzliche Sub-Domains stehen. Als Trennzeichen zwischen den einzelnen Namensteilen wird ein einfacher Punkt benutzt.

Ein kompletter Rechnername sieht also z.B. so aus: **trillian.de.wusel.net**

Dabei ist **trillian** der eigentliche *Hostname*, während das zwischengeschobene **de** eine Subdomain beschreibt. Die *2<sup>nd</sup> Level-Domain* ist hier **wusel**, während **net** die Top Level-Domain ist.

Diese Unterteilung des Namens hat keine optische, sondern rein organisatorische Gründe: an jedem Trennpunkt kann die administrative Verantwortung verlagert werden. Dadurch ist das DNS eine weltweit verteilte Datenbankanwendung, die über keine zentrale Verwaltungsstelle aller Daten verfügt. Jeder Nameserver kann einen Teil des Namensraumes eigenverantwortlich verwalten.

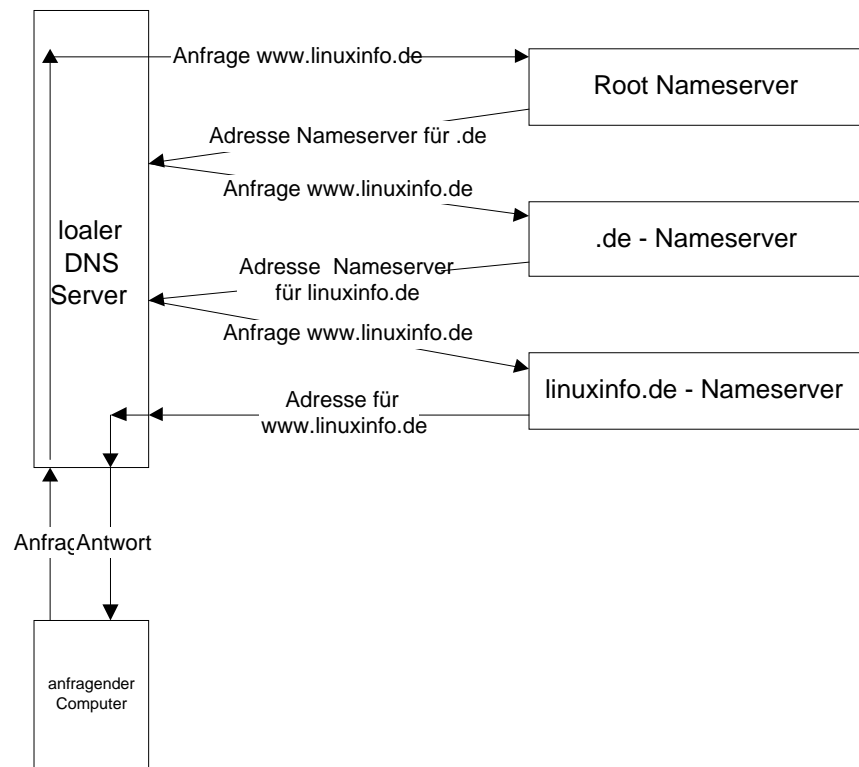
Die *Topleveldomains* werden von zentralen Stellen verwaltet, *Network Information Centers* oder *NIC* genannt, die wiederum *2<sup>nd</sup> Level-Domains* unterhalb ihrer zuständigen Toplevel Domain vergeben bzw. die Vergabe an andere

Nameserver delegieren. Derzeit gibt es folgende sieben generische Top Level Domains: **com**, **org**, **net**, **edu**, **int**, **mil**, **arpa**. Daneben gibt es für jedes Land eine eigene Top Level Domain, die dem ISO - Code des jeweiligen Landes entspricht – für Deutschland also **.de**

Die Nameserver werden entsprechend dieser Unterteilung der Namen-Bereiche strukturiert. Auf der obersten Ebene finden sich die sogenannten *Root-Nameserver*, die Informationen

über die für die jeweilige Top Level-Domain zuständigen Nameserver bereit halten. Entsprechend verfügen diese über Informationen zu den Nameservern der untergeordneten Domainteile – und so weiter, bis ein Nameserver die Information über den gewünschten Hostnamen anbietet.

Die Auflösung eines kompletten Systemnamens, d.h. seine Zuordnung zu einer IP Adresse, könnte so aussehen:



Hier wird nach der IP-Adresse des Rechners **www.linuxinfo.de** gesucht. Der suchende Rechner richtet seine Anfrage an den in ihm eingestellten DNS-Server. Dieser leitet sie an die Root-Nameserver weiter, die die Adresse des für die Toplevel-Domain **.de** zuständigen DNS-Servers mitteilen. Die darauf folgende Anfrage bei diesem System retourniert dem fragenden Computer die Adresse des für die Domain **linuxinfo.de** verantwortlichen Nameservers. Eine weitere Anfrage bei dieser Adresse ergibt die Adresse des Rechners **www.linuxinfo.de**.

Diesen Vorgang, d.h. die Suche nach einem Namen oder einer IP-Adresse, nennt man *Resolving*, also Auflösung. Sucht man bei gegebenem Namen nach einer Adresse, wird das als *forward*

*resolving* bezeichnet – den umgekehrten Vorgang, d.h. die Suche nach dem Namen bei bekannter IP-Adresse, nennt man *reverse resolving*.

Will man das *reverse resolving* besser verstehen, sind weitere Informationen über die Organisation des Adressraumes im DNS nützlich. Alle Adressen sind im DNS in der Subdomain *in-addr* der Top Level Domain **arpa** eingetragen. Unterhalb dieser Zone werden die Adressen in umgekehrter Notation aufgeführt. So erscheint die Adresse **212.14.66.4** in der Zone **in-addr.arpa** als **4.66.14.212.in-addr.arpa**. Wird nun ein entsprechendes Resolving gestartet, konvertiert der fragende Computer die Adresse automatisch in die entsprechende Notation. Anschließend läuft der Auflösungsprozess wie beschrieben ab.

Um die Funktionalität des DNS sicherzustellen, auch wenn ein Server ausfällt, sind die Daten einer Domain oder Sub-Domain grundsätzlich mehreren Servern bekannt – einem *primären* und wenigstens einem *sekundären* DNS-Server. Der letztere gleicht sich regelmäßig mit dem primären Server ab und übernimmt dessen Daten. Alle für einen Bereich zuständigen Nameserver sind beim übergeordneten Nameserver als autoritativ für ihren Bereich registriert. Man spricht von einer DNS-Delegation zu diesen Servern, da nur sie die für diese Domain glaubwürdigen Daten haben.

Die Namen- und Adressdaten werden auf dem primären DNS-Server editiert. Der oder die sekundären Server übernehmen sie dann. Die Zuordnung als primärer und sekundärer Server erfolgt seitens des jeweiligen Systemverwalters. Die übergeordneten Server interessiert diese Zuordnung nicht, da sie alle eingetragenen DNS-Server als gleichberechtigt ansehen.

## BIND - die Standardsoftware

Die Standard-Nameserversoftware unter Unix – und natürlich unter Linux – ist BIND. Die Software ist frei verfügbar, wird vom Internet Software Consortium *ISC* gepflegt und ist bei dessen Webserver erhältlich: <http://www.isc.org/>.

Die Grundkonfiguration des BIND erfolgt über eine ASCII-Datei namens `NAMED.CONF`, die im allgemeinen im Verzeichnis `/ETC` liegt. Eine übliche Grundkonfiguration sieht so aus:

```
options {
    directory "/var/named";
    listen-on {
        212.14.66.4;
        127.0.0.1;
    };
    query-source address 212.14.66.4;
    interface-interval 5;
    statistics-interval 0;
    transfers-in 20;
    transfers-per-ns 5;
};
zone "." in {
    type hint;
    file "root.cache";
};
```

Die wichtigsten Elemente dieser Konfiguration sind die Statements **directory**, das das Datenverzeichnis für BIND bezeichnet, und **listen-on**, das die Adressen definiert, über die der Nameserver erricht wird, also an welche sich der Prozess bindet. Hier können mehrere IP-Adressen durch einen Strichpunkt getrennt angegeben werden.

Ein weiterer wichtiger Eintrag ist die Definition der *root-Cache-Datei*, die nach dem Stichwort **zone "."** definiert wird. In dieser Datei werden die Adressen der Root-Nameserver hinterlegt. Bis auf den Parameter **hint** der **type**-Option handelt es sich hier um eine ganz normale Zonendefinition. Dabei beschreibt der Parameter der Option **file** die Datei, in welcher die Rootserverdaten hinterlegt sind – eben die *Root Cache-Datei*.

Auch die *Zone* oder *Zonendatei* muss konfiguriert werden, also die Datei, in der die Daten für eine Domain oder Sub-Domain eingetragen sind. Diese Definition findet mittels des **zone**-Parameters innerhalb der Datei `NAMED.CONF` statt. Dabei wird als erster Parameter der Domainname angegeben, dann der Typ – also das Schlüsselwort **master** bei einer primären bzw. **slave** bei einer sekundären Zone. Endlich wird der Option **file**, wie oben erwähnt, der Name der Zonendatei übergeben. Den detaillierten Aufbau von Zonendateien sehen wir uns nachfolgend an.

Das folgende Beispiel zeigt die Deklaration einer primären und einer sekundären Zone:

```
zone "wusel.net" in {
    type master;
    file "prim/wusel.net";
};
zone "rkcom.net" in {
    type slave;
    file "sec/rkcom.net";
    masters {
        212.14.66.1;
    };
};
```

Wie ersichtlich muss bei der Bekanntgabe einer sekundären Zone die IP-Adresse des zugehörigen **master** angegeben werden, also die Adresse des Nameservers, der diese Domain als primäre Zone fährt, von dem also die entsprechenden Daten übertragen werden.

**Aufbau von DNS-Zonendateien**

Die Inhalte einer Domain werden innerhalb der DNS-Zonendateien verwaltet. Dies bedeutet, daß hier die Zuordnung von Namen zu IP-Adressen und von IP Adressen zu Namen erfolgt. Eine solche Zuordnung oder *Resource Record*) sieht wie folgt aus:

**<NAME> <CLASS> <Type> <RData>**

Innerhalb des DNS kennt man folgende Klassen:

<i>Klasse</i>	<i>Bedeutung</i>
<b>IN</b>	Internet
<b>CS</b>	CSNET
<b>CH</b>	CHAOSnet
<b>HS</b>	Hesiod

Heute hat nur noch IN-Internet Bedeutung.

Das Type - Feld beschreibt die Art des Eintrages. Hier gibt es folgende Typen:

<i>Type</i>	<i>folgende Daten (Rdata)</i>	<i>Bedeutung</i>
<b>A</b>	zum Namen gehörende IP-Adresse	ordnet einem Namen eine IP-Adresse zu
<b>NS</b>	Name eines zugehörigen Nameservers	delegiert eine Subdomain heraus. Dabei sind die hier angegebenen Rechner für diese Subdomain als DNS-Server definiert.
<b>CNAME</b>	zu einem Alias gehöriger Name	definiert einen Aliasnamen für einen bereits existieren Namen, der mittels eines A Records einer IP-Adresse zugeordnet sein muss
<b>PTR</b>	Zeiger auf einen Namen (Pointer)	ordnet einer IP-Adresse einen DNS-Namen zu. Dieser sollte wiederum mittels eine A-Records auf die gleiche Adresse zeigen. Hier wird das sogenannte <i>Reverse Mapping</i> definiert.
<b>HINFO</b>	Hardwareinforma-tion für den Eintrag	enthält Informationen über die Hardware der vorangehenden Namen.
<b>MX</b>	Mailexchange-system (Mailserver)	enthält Priorität und DNS-Namen des Mailservers, der für diesen Namen Mail entgegennimmt. Dieser Name muß mittels eines A-Records einer IP-Adresse zugewiesen sein.
<b>TXT</b>	Texteintrag	gibt einen zum Eintrag gehörenden Text an. Dies kann eine Bemerkung zum entsprechenden System sein. Der Text sollte in doppelten Anführungszeichen stehen.

Neben den einzelnen Datensätzen benötigt jede Zone einige Verwaltungsdaten. Diese sind in einer bestimmten Sektion am Anfang jeder DNS-Zone im sogenannten *Start of Area* oder *SOA*) definiert. Genaugenommen handelt es hier nur um einen gesonderten Typus eines DNS-Records.

Ein üblicher SOA-Record besitzt folgenden Aufbau:

```
<domainname> IN SOA
<primärer_Nameserver>
<zonen_administrator> (
  <serial>
  <refresh>
  <retry>
  <expire>
  <TTL>
)
```

Die Angaben haben folgende Bedeutung:

- **Primärer\_Nameserver**  
Name des primären Nameservers
- **zonen\_administrator**  
Emailadresse des Zonen-Administrators. Statt des @ - Zeichens steht hier ein Punkt.
- **seriel**  
Seriennummer der Zonnendatei. Anhand der Seriennummer erkennt ein sekundärer Server eine Veränderung beim primären Server. Ist die Seriennummer beim primären Server höher als in der Datei des sekundären Servers, kopiert dieser erneut die Zonendatei. Das bedeutet, daß die Seriennummer bei jeder Änderung an der Zonendatei erhöht werden muß. Eine Seriennummer darf niemals niedriger werden, da das die sekundären Systeme nicht mitbekommen. Um dies zu verhindern, hat sich in der Praxis folgendes Format für die Seriennummer durchgesetzt:

```
YYYYMMDDNN
```

Dabei bedeuten:

<i>Eintrag</i>	<i>Bedeutung</i>
<b>YYYY</b>	Vierstellige Jahreszahl
<b>MM</b>	zweistellige Monatszahl
<b>DD</b>	zweistellige Tageszahl
<b>NN</b>	Nummer der Änderung am jeweiligen Tag

- **refresh**  
Hier wird der Zeitraum in Sekunden definiert, in dem ein sekundärer Server Seriennummern vergleichen und die Zonendatei ggf. übertragen soll. Ein gängiger Wert hier ist **86400**.
- **retry**  
Zeitwert in Sekunden, innerhalb dessen ein sekundärer Server einen erneuten Verbindungsaufbau probiert, wenn der Server nicht erreicht wurde – meist 10800 Sekunden.
- **expire**  
definiert, wann ein sekundärer Server eine Zone für ungültig erklärt, nachdem der primäre Server nicht erreicht wurde. Ein praxiserprobter Wert sind 2592000 Sekunden.
- **TTL**  
bezeichnet die *Time to Live*. Gibt – ebenfalls in Sekunden – an, wie lange ein anfragender Client Informationen im lokalen Cache hält.

Sehen wir uns das an einer kleinen Beispielzone an:

```
wusel.de      IN      SOA      ns.wusel.de. hostmaster.nic.wusel.de. (
                199912162      ; Serial
                86400      ; Refresh
                10800      ; Retry
                2592000      ; Expire
                86400      ) ; Minimum

                IN      NS      ns.ISAR.net.
                IN      NS      ns.Essen.ISAR.net.
                IN      MX      100 mail.wusel.de.
                IN      MX      110 relay.wusel.de.
www           IN      A      212.14.66.4
mail         IN      A      212.14.66.4
relay        IN      A      212.14.66.44
web          IN      CNAME   www.wusel.de.
sls          IN      NS      ns.wusel.de.
ns           IN      A      212.14.66.4
```

## Der Aufbau von Reverse Zones

Der Aufbau einer *Reverse Zone* entspricht der einer *Forward Zone* – der einzige Unterschied

ist, dass links anstelle eines Namens die IP-Adresse steht und hinter dem Typekennzeichen *PTR* der volle Rechnername.

```
@      IN      SOA      ns.wusel.de. hostmaster.nic.wusel.de. (
                                199912162      ; Serial
                                86400      ; Refresh
                                10800      ; Retry
                                2592000      ; Expire
                                86400      ) ; Minimum
                                IN      NS      ns.ISAR.net.
                                IN      NS      ns.Essen.ISAR.net.
0      IN      PTR      net.wusel.de.
1      IN      PTR      karlsfeld.de.wusel.net.
2      IN      PTR      hades.wusel.de.
...
```

## Kontrolle des Nameservers

Für die Kontrolle des Nameserverprozesses enthält die Distribution des BIND ein kleines Steuerprogramm namens **ndc** oder **named control**, das den **named** kontrolliert. Die wichtigsten Parameter folgen hier:

- **status**  
zeigt den aktuellen Status des **named** an.
- **reload [Zonendatei]**  
lädt alle oder die angegebene Zone neu in den Speicher des Nameservers
- **start**  
startet den Nameserverprozess
- **stop**  
beendet den laufenden Nameserverprozess

- **restart**  
startet einen neuen Nameserverprozess neu
- **trace**  
versetzt den **named**- in den **trace**-Modus. Damit protokolliert der Prozess alle Anfragen in die Datei **named.run**. Der **Trace**-Modus muss unbedingt wieder ausgeschaltet werden, da diese Datei sehr schnell sehr gross werden und die Platte bis zum Rand füllen kann.
- **notrace**  
beendet den **trace**-Modus des **named**.

Weitere Informationen zum BIND finden sich unter der Adresse <http://www.isc.org/>.

## Mailserver

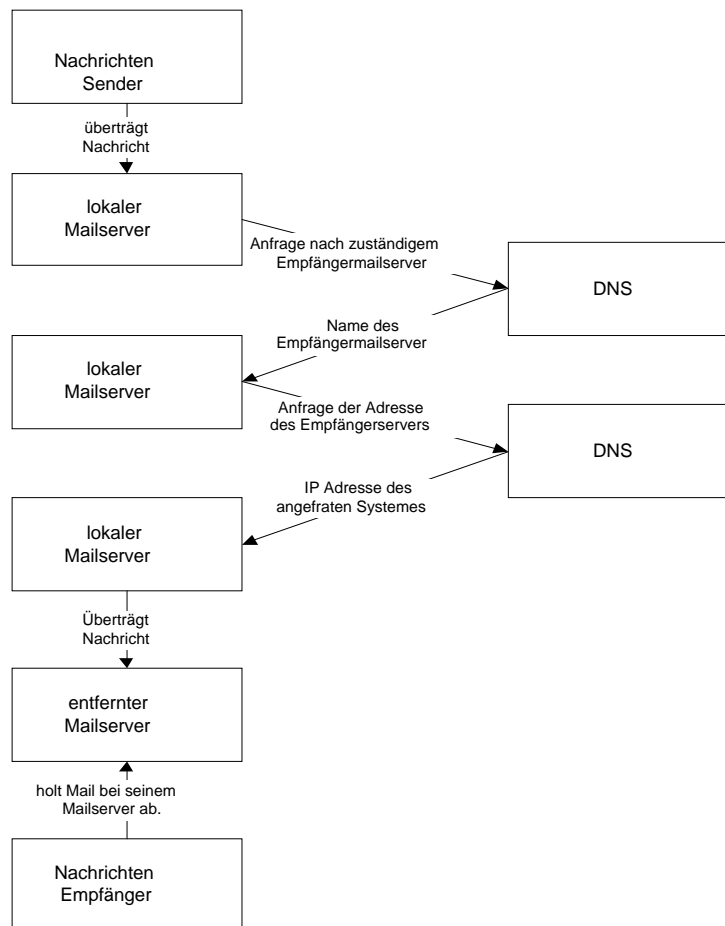
Email ist einer der ältesten und meistgenutzten Dienste im Internet. Mit seiner Hilfe werden Nachrichten zwischen Benutzern im Netzwerk bzw. dem Internet ausgetauscht. Die aufgabengerechte Betreuung eines Mailservers erfordert einige Hintergrundinformationen zu Funktionsweisen und Protokollen von Email.

Der Versand von elektronischer Post in Internet erfolgt wie schon auf Seite 25 erwähnt mittels des *Simple Mail Transport Protocol*, des *SMTP*. Der Mailsender liefert eine Nachricht bei seinem Mailserver ab, der dann den weiteren Versand bis zum Mailserver des Nachrichtempfängers übernimmt. Dabei benutzt er den DNS-Dienst, der die Informationen über den Mailserver des Empfängers im sogenannten *MX-Record* zur Verfügung stellt. Dieser Eintrag definiert den

Mailserver, der für die Zieldomain eingehende Mail verwaltet. Der sendende Server richtet also zunächst eine Anfrage an DNS bezüglich des zuständigen Mailservers und stellt diesem dann die Nachricht zu. Der Empfängerserver nimmt die Nachricht entgegen und speichert sie, bis der Empfänger die Nachricht abholt.

Das *Abholen* der Mail durch ihren Empfänger erfolgt nicht etwa über das *SMTP*-Protokoll, sondern mittels *POP3* bzw. *IMAP*. Diese Dienste sind meist nicht ein integraler Bestandteil des Mailservers, sondern eigene Dienste und Serverprozesse.

Schematisch sieht die Sache etwa so aus:



Befassen wir uns nun mit der Mailserverseite – ohne Mailserver ist grundsätzlich keine Nachrichtenübermittlung möglich. Der unter Linux wohl meist eingesetzte Mailserver ist **sendmail**. Wie die anderen Programme ist auch er freie Software – im Internet ist er abrufbar unter der Adresse <http://www.sendmail.org/>

Wie die meisten Serverprogramme unter Unix wird **sendmail** mittels einiger Textdateien konfiguriert. Die zentrale Konfigurationsdatei ist die Datei `/ETC/SENDMAIL.CF`. Da diese Datei

sehr komplex und – zumindest für das menschliche Auge – nur schwer lesbar ist, wird sie im allgemeinen mit Hilfe einer Makrosprache generiert – und zwar der Makrosprache **m4**. Durch die Aneinanderreihung der relevanten **m4**-Makros und einer anschließenden Übersetzung in das **cf**-Dateiformat vermag der Administrator eines Mailservers eine Konfiguration relativ einfach zu erstellen.

Eine Standardkonfiguration als **m4**-Makro sieht so aus:

```
VERSIONID(`lnx-as-1.0')

OSTYPE(linux)

MASQUERADE_AS(domain.de) dnl # lokale Mail maskieren

define(`confCF_VERSION', `lnx-as-1.0')dnl # Versionszusatz
define(`confERROR_MODE', m)dnl # Errormodus setzen
define(`confQUEUE_LA', 10)dnl # Ab Load 10 nur noch queueing
define(`confREFUSE_LA', 20)dnl # Ab Load 20 mail zurückweisen
define(`confPRIVACY_FLAGS',
`needmailhelo,noexpn,needvrfyhelo,noreceipts,authwar
nings')dnl # EinigeSicherheitseinstellungen

FEATURE(redirect)dnl # Redirect Feature aktivieren
FEATURE(use_cw_file) dnl # CW File f. lokale Domains nutzen
FEATURE(relay_entire_domain) dnl
FEATURE(masquerade_envelope) dnl

MAILER(smtp)
```

Soll diese Datei das richtige Format erhalten, wird sie mittels des Befehls **m4** konvertiert – über folgenden Aufruf:

```
m4 ./sendmail.mc > /etc/sendmail.cf
```

Damit wären nun die Basiseinstellungen für **sendmail** vorgenommen, und der Versand von Mail ist grundsätzlich möglich. Die Anpassung der Konfiguration an das eigene System erfordert einige Feinarbeit. Zunächst müssen in der Datei `/ETC/SENDMAIL.CW` alle Domains eingetragen werden, die der Mailserver als lokale Domains behandeln soll. Dabei werden die Domains einfach untereinander eingetragen:

```
wusel.de
wusel.com
wusel.net
```

Fehlt noch die Konfiguration für den Empfang von Email. Sollen lokalen Benutzern mehrere Emailadressen zur Verfügung stehen, nimmt man in der Datei `/ETC/ALIASES` die entsprechenden Zuordnungen vor. Die Syntax ist einfach:

```
Alias:      lokaler Benutzer
```

Als Beispiel folgt ein Ausschnitt aus einer `/ETC/ALIASES`:

```
Achim.Schmidt:  schmidt
achim:          schmidt
as:             schmidt
```

Damit wird Mail für *Achim.Schmidt*, für *achim* und für *as* beim lokalen Benutzer *schmidt* abgeliefert. Aktiviert wird diese Datei mit dem Befehl **newaliases**.

Ein weiterer wichtiger Punkt bei einem Mailserver ist die Sicherheit des Servers vor *SPAM-Relay*-Angriffen. Ein solcher Angriff veranlasst die Weitersendung einer Mail von außen an viele Adressen ausserhalb des lokalen Netzwerks.

Solche Angriffe werden mit Hilfe der Datei [/ETC/MAIL/ACCESS](#) abgewehrt, deren Benutzung in der obigen Beispielkonfiguration enthalten ist. Hier werden die Domains bzw. IP-Adressen eingetragen, die diesen Server als Relay-Server benutzen, also Mail einliefern dürfen. Einen entsprechender Eintrag zeigt das folgende Beispiel:

```
192.168.1.          RELAY
```

Ist die Datei erstellt, muss sie ins DB-Format übersetzt werden – mit dem Befehl:

```
makemap hash access < access
```

**sendmail** ist ein sehr potenter Mailserver. Es würde jedoch den Rahmen dieses Buches sprengen, wenn wir hier auf alle seine Möglichkeiten eingehen. Es gibt aber gute Fachliteratur zu diesem Programm, und auf der Webseite des Programms finden sich weitere Informationen unter folgender Adresse:

<http://www.sendmail.org/>

## Dienste für Windowsclients - Samba

Linux kann Datei- und Druckerdienste auch für Windowsclients verfügbar machen – mit Hilfe des Pakets *Samba*, das hier abrufbar ist:

<http://de.samba.org/samba/samba.html>

*Samba* bietet auch viele Dienste an, die normalerweise nur von *Windows NT* erbracht werden – z.B. folgende:

- WINS-Server
- Computersuchdienst
- Loginserver
- Primärer Domain-Server
- Diagnosewerkzeuge

Gegenüber herkömmlichen NT-Servern besitzt *Samba* die Vorteile des darunterliegenden Serverbetriebssystems, z.B. Linux – als da u.a. wären:

- standortunabhängige Administration
- Stabilität
- zentrale Konfiguration

Die Kommunikation innerhalb von Windowsnetzen erfolgt normalerweise mittels des *NETBIOS*-Protokolls. Im Kern setzt sich dieses Protokoll aus drei Hauptbestandteilen zusammen:

- **Namensdienst**  
sorgt für die Identifikation der angeschlossenen Stationen und ist Grundlage des Computersuchdienstes.
- **Datagrammservice**  
regelt die Kommunikation zwischen den einzelnen Rechnern. Der Datenaustausch erfolgt dabei in einzelnen Paketen, den sogenannten Datagrammen.
- **Sitzungsdienst**  
sorgt für die fehlerfreie Kommunikation der Netzstationen.

Das NetBIOS-Protokoll kann in verschiedenen anderen Protokollen implementiert werden. Die derzeit bekanntesten sind *NetBEUI*, *IPX* und natürlich *TCP/IP*. *Samba* benutzt die *TCP/IP* Implementierung – und somit die Vorteile dieses Protokolles.

Das *Samba* Paket besteht aus folgenden Teilen:

```
swat      stream tcp      nowait.400      root /usr/sbin/swat swat
```

- **SMBD**  
zentraler Serverdienst für die Datei- und Druckservices
- **NMBD**  
Daemon für Namens- und Datagrammdienste. Das Programm reserviert beim Start von *Samba* die entsprechenden NetBIOS-Namen; es ist der WINS-Server und somit zuständig für die Namensauflösung.
- **TESTPARM**  
prüft die Sambakonfigurationsdatei auf syntaktische Richtigkeit
- **SMBPASSWD**  
pflegt die verschlüsselten Passwörter der Serverseite.
- **SMBCLIENT**  
ermöglicht dem lokalen Linuxrechner den Zugriff auf Ressourcen, die von NT Servern freigegeben wurden.
- **NMBLOOKUP**  
Diagnosewerkzeug zur NetBIOS-Namensauflösung.
- **/ETC/SMB.CONF**  
zentrale Sambakonfigurationsdatei des Linuxservers.
- **/ETC/SMBPASSWD**  
für *Samba* genutzte Passwortdatenbank
- **SWAT**  
webbasiertes Administrationswerkzeug für *Samba*.

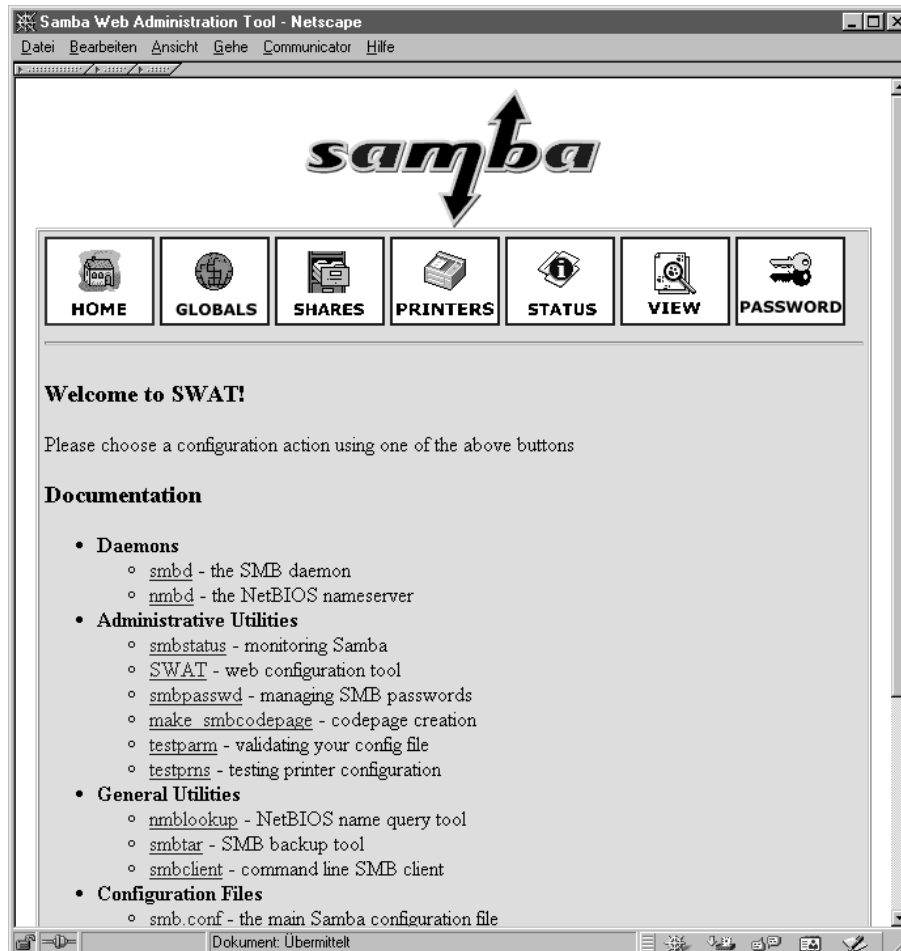
### Basiskonfiguration von Samba

*Samba* kann auf zwei verschiedene Weisen konfiguriert werden: zum einen durch die manuelle Bearbeitung der Datei */ETC/SMB.CONF*, zum andern über das webbasierte Konfigurationswerkzeug *swat*.

Da das Editieren der Konfigurationsdatei sehr mühsam ist, wenden wir uns lieber gleich *swat* zu. Dieses Programm erlaubt die komplette Konfiguration des Sambadienstes. Gestartet wird *swat* normalerweise mittels des *inetd*-Superservers, was folgenden Eintrag in dessen Konfigurationsdatei */ETC/INETD.CONF* erfordert:

**swat** läuft daraufhin als eigener Webserver auf Port 901 des Samba-servers und kann mittels eines normalen Webbrowsers dort angesprochen werden. Nach der ersten Verbindung muss man sich beim System anmelden, und zwar als User **root** mit dem System-Passwort dieses Users.

Nach erfolgreicher Authentifikation erscheint das Hauptmenü, das die verschiedenen Funktionen des Programms sowie Links zur Dokumentation anbietet.



Im Menüpunkt **GLOBALS** werden die zentralen Konfigurationen für den Server vorgenommen. Diese gliedern sich in verschiedene Gruppen:

- **BASE OPTIONS** (Basiseinstellungen)
- **SECURITY OPTIONS** (Sicherheitseinstellungen)
- **LOGGING OPTIONS** (Protokolleinstellungen)
- **TUNING OPTIONS** (Performanceeinstellungen)
- **BROWSE OPTIONS** (Anzeigeeinstellungen)
- **WINS OPTIONS** (WINS-Einstellungen)

Innerhalb dieser Gruppen werden Konfigurationsparameter für den späteren Betrieb des Samba-servers gesetzt. Die wichtigsten sind:

- **WORKGROUP**  
Angabe der Windows-Arbeitsgruppe oder Windows NT-Domain, der der Samba-server angehören soll.
- **NETBIOS NAME**  
Name des Samba-servers im Windows-Netzwerk.
- **SERVER STRING**  
Namensergänzung, die innerhalb der Windows-Netzwerksicht angezeigt wird.
- **INTERFACES**  
Angabe der Interfaces, die *Samba* benutzen soll. Dabei wird die IP-Adresse des Samba-servers auf dem entsprechenden Netzinterface und dessen Netzmaske

angegeben – z.B. **192.168.1.2/24** oder **192.168.1.2/255.255.255.0**. Sollen mehrere Interfaces genutzt werden, kann man mehrere Kombinationen aus IP-Adresse und Netzmaske angeben, die jeweils durch ein Leerzeichen getrennt werden.

- **SECURITY**

legt den Zeitpunkt der Identifikation fest. Hier sind mehrere Möglichkeiten denkbar:

**share:** legt fest, dass die Prüfung beim Ansprechen einer Freigabe stattfindet

**user:** prüft das Passwort bereits bei der Herstellung einer Benutzersession zum Server.

**server:** Bei dieser Konfiguration prüft der Samba-Server die Kombination aus Username und Passwort, indem er eine Verbindung zu einem anderen Server aufbaut und versucht sich mit den erhaltenen Daten dort anzumelden. Ist dieser Vorgang positiv, gibt er dem Request statt; schlägt die Anmeldung fehl, weist der Samba-Server die aufzubauende Session ab. Um diese Funktion zu nutzen, wird noch die Option **password server** benötigt, welche den NetBIOS-Namen des Servers angibt, bei dem die Authentifikation stattfindet.

**domain:** Der hier eingesetzte Mechanismus funktioniert ähnlich dem der Option **server**. Der Unterschied liegt darin, dass hier eine Windowsauthentifikation gegen einen Domainserver stattfindet. Um diese Option zu nutzen, muß aber ein Computerkonto in der entsprechenden Domain existieren, und der Samba-Server muß über ein entsprechendes Passwort verfügen, das mittels **smbpasswd -j <DOMAIN> -r <PDC>** generiert wird.

- **ENCRYPT PASSWORDS**

gibt an, ob verschlüsselte Passwörter benutzt werden sollen. Windows NT 4.0 ab SP3 und Windows 98 haben dies als Standard.

- **HOSTS ALLOW/HOSTS DENY**

Hier können Netze oder einzelne Systeme eingetragen werden (IP Adressen), denen der Zugriff auf den Server erlaubt oder verboten wird.

Einzelne Dateifreigaben werden im Menüpunkt **SHARES** geregelt, wo neue Freigaben angelegt werden und der Zugriff auf verwaltet wird. Dabei gibt es folgende Einstellungsmöglichkeiten:

- **BASE OPTIONS**

- **comment:** Name der Freigabe

- **path:** Pfad zum Verzeichnis auf Linuxebene

- **SECURITY OPTIONS**

Sicherheitseinstellungen. Regelung der Zugriffsrechte auf dieses Verzeichnis und Auflistung der berechtigten/nicht-berechtigten IP-Adressen.

- **BROWSE OPTIONS**

Angabe, ob die Freigabe in der Liste der Freigaben dieses Servers angezeigt wird oder nicht.

- **MISCELLANEOUS OPTIONS**

**available:** gibt an, ob diese Freigabe verfügbar ist oder nicht.

Im Menüpunkt **PRINTERS** werden die gleichen Einstellungen für Druckerfreigaben definiert.

Der nächste wesentliche Punkt ist der Menüpunkt **STATUS**. Hier wird der Betrieb des Samba-Servers kontrolliert – so wird er hier gestartet, gestoppt oder erneut gestartet. Außerdem werden hier einige Statusinformationen wie z.B. die aktiven Verbindungen angezeigt.

Im **PASSWORD**-Menü können Einstellungen für Sambabeanutzer vorgenommen werden.

Allerdings müssen alle diese Sambabeanutzer als lokale User auf der Linuxmaschine existieren.

## Sicherheit in Linuxnetzen

Sicherheit von Linuxservern und im Netz ist ein viel beachtetes Thema, gerade bei Netzen, die über einen Anschluß an öffentliche Netze wie das Internet verfügen. Ein weit verbreitetes System wie das Netzwerkbetriebssystem Linux ist oft Ziel von Angriffen und Einbruchsversuchen. Man muß dabei jedoch unterscheiden, ob es sich um einen „räuberischen“ Einbruchsversuch handelt oder um einen Angriff, der den Rechner nur lahmzulegen versucht, z.B. durch sogenannte *Denial of Service*- oder *DoS*-Angriffe.

Bei letzteren Angriffen geht es schlicht darum, den Zielrechner lahmzulegen. Dies erfolgt meist durch ein Überfluten mit Anfragen. So versucht man einen Webserver durch das zielgerichtete und schnelle gleichzeitige Senden von HTTP-Anfragen zu überlasten – oder man sendet viele Pakete an den SMTP Port eines Mailservers.

Auf der anderen Seite stehen Attacks, bei denen der Angreifer in den entfernten Rechner einzudringen versucht, um dessen Dienstleistungen zu nutzen. Hier muss der Angreifer viel gezielter vorgehen und Sicherheitslücken im Betriebssystem oder Leichtfertigkeiten des Verwalters ausnutzen. Fehler im Betriebssystem gibt es in jedem System – und immer wieder. Hier liegt es am Administrator, sich über auftauchende Fehlerquellen zu informieren und entsprechende Gegenmaßnahmen einzuleiten. Eine gute Möglichkeit, sich auf dem Laufenden zu halten, sind Mailinglisten der Distributoren oder anderer Einrichtungen, die sich mit der Sicherheitsthematik beschäftigen, wie z.B. *CERT*. Natürlich gibt es auch im World Wide Web entsprechende Services und Übersichten – so etwa die *Security News Base* von LinuxInfo.DE, <http://www.linuxinfo.de/>.

Die andere große Gefahrenquelle ist die Leichtfertigkeit von System- und Netzadministratoren. Jedes Betriebssystem ist nur so gut und sicher wie sein Verwalter es zulässt. So sollten Dienste, die nicht benötigt werden, gar nicht angeboten und entsprechende Prozesse gar nicht gestartet werden. Dienste, die nur von bestimmten Stationen benutzt werden, sollten nur für diese bereit stehen. Hier ist ein *TCP-Wrapper* eine gute Möglichkeit, Serverdienste nur bestimmten Stationen verfügbar zu machen – später dazu mehr.

Neben Angriffen von außen darf man auch die internen Gefahren nicht unterschätzen. Die meisten Angriffe auf ein System kommen aus dem eigenen Netz heraus und nicht von außen!

Sehen wir uns zunächst einmal die einfachen Möglichkeiten an, einen Linuxrechner sicherer zu machen und Angreifern auf diese Weise nicht allzu einfache Angriffsflächen zu bieten.

### Was man nicht anbieten will, braucht nicht zu laufen

Dies ist einer der Leitsätze, an die sich jeder Administrator grundsätzlich halten sollte. Denn Prozesse, die nicht laufen, bieten keine Angriffsflächen. Darum sollte man genau überlegen, welche Dienste man auf einem Rechner anbieten will, und alle anderen evtl. installierten Dienste abschalten – entweder indem man die *INIT-Scripts* entsprechend modifiziert oder indem man die *inetd*-Konfigurationsdatei `/etc/inetd.conf` anpasst.

So banal sich dies liest, so erschreckend ist, welche Unmengen an Diensten viele Systeme anbieten, ohne daß sich der Systemverwalter darüber eigentlich im klaren ist. Und eben solche Services sind beliebte Einbruchsstellen, da sie meist auch nicht weiter beachtet und gepflegt werden.

### Dienste reglementieren

Viele Dienste auf Linuxservern sollten nur dem lokalen Netzwerk oder bestimmten Rechnern im Internet zur Verfügung stehen. Hier sind diese Dienste so weit zu regulieren, daß nur die relevanten Stationen Zugriff haben. Eine Beschränkung via Username und Password ist dabei nicht ausreichend – um diese Identifikation zu starten, wird nämlich das entsprechende Programm angeboten und gestartet. Ist in diesem Prozess ein Fehler, kann dieser evtl. auch ohne Zugriff ausgenutzt werden, um unbefugte Rechte zu erlangen. Besser ist es hier, auf einer tieferen Ebene anzusetzen, und durch ein vorgeschaltetes Programm zu prüfen, ob die Dienstanforderung von der berechtigten Adresse kommt – und nur in diesem Falle den entsprechenden Service zu starten. Selbstverständlich kann eine Useranmeldung zusätzlich eingesetzt werden. Sollte das Programm nun einen Fehler haben, wird es für

Anforderungen von nicht erlaubten Adressen erst gar nicht gestartet, und derartige Sicherheitslücken können nicht ausgenutzt werden.

Ein Programm, das man solchen Diensten vor-schalten kann und das die Adresse der Anforderung prüft, ist der *TCP-Wrapper*. Bei den meisten Linuxdistributionen ist er als Standard installiert. Der TCP-Wrapper kann auch Dienste absichern, welche mittels des `inetd` gestartet werden. Dazu wird in der `/ETC/INETD.CONF` nicht der eigentliche Server gestartet, sondern der TCP-Wrapper – meist `tcpd`. Als Argument wird ihm der zu startende Prozess übergeben, worauf er anhand seiner Konfigurationsdateien nachprüft, ob eine anfragende IP-Adresse für den angeforderten Dienst Zugriffsrechte hat oder nicht.

Diese Dateien sind die `/ETC/HOSTS.ALLOW` und die `/ETC/HOSTS.DENY`. Hier werden die Zugriffsrechte hinterlegt. Diese Zuordnung findet pro angebotenen Dienst statt. Die `/ETC/HOSTS.ALLOW` enthält die Systeme, denen der Zugriff auf bestimmte Dienste offen steht, während die `/ETC/HOSTS.DENY` die Systeme enthält, die keinen Zugriff auf bestimmte Services haben sollen.

In der `/ETC/HOSTS.ALLOW` werden die Dienste eingetragen, auf die bestimmte Systeme Zugriff haben sollen. Dabei wird erst der Dienst genannt, dann die Systeme, die Zugriffsrechte haben. Als Trennzeichen zwischen Service und Systemen wird ein Doppelpunkt benutzt. Um die Systeme aufzulisten, können einzelne DNS-Namen, DNS Fragmente – wie z.B. `.wusel.de` für alle Rechner, deren DNS Name auf `.wusel.de` endet – oder aber IP-Adressen eingesetzt werden. Sollen IP-Netze gelistet werden, kann man dies in der Form Netzadresse/Netzmaske angeben – z.B. `192.168.1.0/255.255.255.0`. Sollen mehrere Systeme angegeben werden, kommt ein Leerzeichen als Trenner zum Einsatz.

Das Wort `ALL` ist ein spezielles Schlüsselwort, das sowohl alle Dienste als auch alle Rechner bezeichnen kann.

Eine `/ETC/HOSTS.ALLOW` sieht so aus:

```
in.telnetd: .wusel.de
sshd: ALL
ipop3d: .wusel.de .wusel.net
```

... und eine `/ETC/HOSTS.DENY` so:

```
ALL: ALL
```

In den obigen Beispielen dürfen demnach auf den *Telnet*service nur Rechner aus `wusel.de` zugreifen, während alle Systeme mittels `SSH` Verbindung aufnehmen dürfen. Auf den POP3-Dienst wiederum dürfen nur Rechner aus den Domains `wusel.de` und `wusel.net` zugreifen. Da in der `/ETC/HOSTS.DENY` nur `ALL: ALL` steht, sind alle Dienste, die nicht in der `/ETC/HOSTS.ALLOW` aufgeführt sind, verboten. Es werden somit nur die Dienste angeboten, die in der `/ETC/HOSTS.ALLOW` stehen.

Natürlich greifen diese Regulierungen nur, wenn der entsprechende Service auch mittels des `tcpd` gestartet wird oder dessen Konfigurationsdateien wie z.B. `SSH` nutzt. Services, die eigenständig laufen und diese Dateien nicht nutzen, wie etwa der Nameserver `BIND`, werden so natürlich auch nicht reguliert. Hier kann man entsprechende Konfigurationen jedoch meist in der eigenen Konfigurationsdatei der Software treffen. Generell sollte man Dienste nur Systemen ermöglichen, die Zugriffsrechte auf sie haben.

Mit dieser Strategie hat man schon eine große Menge an potentiellen Sicherheitslücken eingedämmt, da man bei sicherheitsrelevanten Dingen nur noch die offenen Services betrachten muß.

## Der Einsatz von Firewalls

Firewalls haben einen großen Vorteil – sie bieten Sicherheit an einer zentralen Stelle und schützen so ein komplettes Netzwerk. Natürlich ist auch hier die Sicherheit nur so gut wie die Qualität der Firewall und der Kenntnisstand des Verwalters.

Sehen wir uns zunächst einmal Firewalltechniken ganz allgemein an – will man die Möglichkeiten verstehen, die Linux bietet, sind grundlegende Informationen über Arten und Funktionsweisen von Firewalls notwendig.

### Paketfiltering

Paketfilterung ist eine simple, aber wirkungsvolle Art, ein Netz zu schützen. Dabei werden alle ein- und ausgehenden Pakete von zentraler Stelle aus betrachtet und aufgrund von Regeln weitergeleitet oder verworfen. Diese Regelsätze können von sehr einfachen und klaren Anweisungen bis hin zu komplexen Regelwerken reichen.

Sinnvollerweise setzt man Paketfilter an einer zentralen Stelle ein, z.B. am Übergang zwischen verschiedenen Netzen. Solche Filter eignen sich also hervorragend für den Einsatz auf *Routern*. Man unterscheidet zwei verschiedene Arten von Filtern, die auf der IP-Adressenebene oder auf der Portebene definiert werden. Der beste Ansatz zum Filtern ist eine Kombination aus Adress- und Portfiltern. Filter dieses Typs sind die meist benutzten und wirkungsvollsten. Hier können Kombinationen von Adressen und Portadressen sowohl als Quell- wie auch als Zieladressen verwendet werden.

Grundsätzlich gibt es zwei Ansätze, Filter zu definieren. Der ersten Variante liegt eine etwas lockere Sicherheitspolitik zu Grunde. Hier ist alles erlaubt, was nicht explizit verboten ist. Dies bedeutet, dass man alle Verbindungen erlaubt und nur einige bestimmte Verbindungen verbietet.

Der zweite Ansatz ist ein sehr restriktiver Ansatz. Hier wird alles verboten, was nicht speziell erlaubt wurde. Diese Variante ist zwar sehr sicher, jedoch sehr komplex zu planen. Denn man muß planen, welche Verbindungen benötigt werden. Das Problem dabei ist die Betrachtung der Folgeverbindungen, wie dies beispielsweise bei FTP der Fall ist.

Wie solche Filter auf Linuxsystemen realisiert werden, sehen wir uns später in Verbindung mit den Möglichkeiten der Implementierung von Firewalltechniken unter Linux an.

### Masquerading oder NAT

Unter *IP-Masquerading* oder *NAT* versteht man die Umsetzung von internen Adressen auf eine einzige Adresse. Dabei werden alle Anfragen des lokalen Netzwerkes vom NAT-System abgefangen und so umgeschrieben, daß ausgehende Pakete nur mit der IP-Adresse des Gateways versendet werden. Das hat den Vorteil, daß in fremden Netzwerken nur die Adresse des Gateways bekannt ist, nicht aber die der internen Maschinen.

### Application Level Gateways

Firewalls, welche auf Applikationsebene arbeiten, gehören zur den sichersten. Bei dieser Technik wird das lokale Netzwerk komplett vom externen Netzwerk getrennt. Die gemeinsame Schnittstelle ist die Firewall, die in beiden Netzen beheimatet ist, jedoch keine direkte Verbindung zwischen den beiden Netzen herstellt. Jegliche Verbindung zwischen den beiden Welten wird über eigene Programme abgearbeitet, sogenannte *Application Gateways* oder *Proxies*. Es handelt sich dabei um Stellvertreter-Dienste, die Anrufe der einen Seite annehmen und sie zunächst prüfen. Ist das Ergebnis positiv, baut der Proxy eine neue Verbindung zum Zielrechner auf und schickt die Anforderung an den eigentlichen Zielhost. Somit ist eine Kommunikation nur über den Stellvertreterprozess möglich – es existiert keine direkte logische Verbindung zwischen Quelle und Ziel. Dabei hat man an der Schnittstelle, der Firewall, alle nur denkbaren Regulierungsmöglichkeiten.

Solche Proxy-Dienste stehen für fast alle Netzanwendungen zur Verfügung. So sind *Application Level Gateways* für WWW, FTP, Telnet, POP3, IMAP, SMTP, Realaudio und – video als freie Software in Internet verfügbar.

Dies ist natürlich nur ein kleiner Einblick in die grundlegenden Firewalltechniken. Es gibt jedoch gute Literatur in diesem Bereich, und auch im Internet gibt es zu diesem Thema eine Fülle an Informationen.

## Firewalltechniken unter Linux

Unter Linux kann man alle drei erläuterten Firewallmechanismen ohne große Probleme implementieren. Dazu bietet zum einen der Kernel viele Möglichkeiten, zum anderen steht entsprechende Software auch als *Open Source*-Software im Internet zur Verfügung.

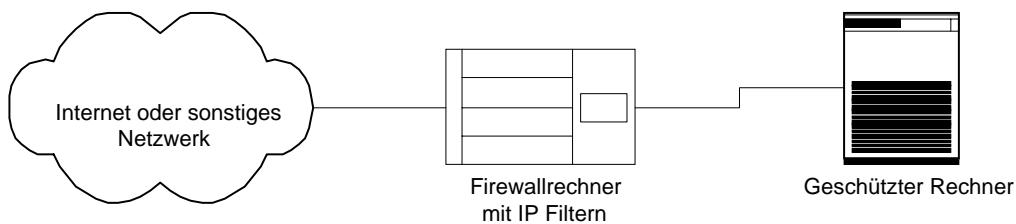
### IP Filter unter Linux

Linux bietet seit dem Kernel 2.0 gute Filtermöglichkeiten. In der Version 2.0 ist das Programmpaket **ipfwadm** enthalten, das eine Filterung ermöglicht. In der Kernelversion 2.2 wurde es durch die deutlich leistungsfähigere **IP CHAINS**-Software abgelöst.

*IP Chains* kennt drei verschiedene Regelarten, auch Ketten oder Chains genannt. Man unterscheidet die **input**-, **output**- und **forward**-Kette. Trifft ein IP-Paket ein, benutzt der Kernel zunächst die input-Kette. Durchläuft das Paket diese Kette erfolgreich, wird es in die nächste Stufe geleitet, die *Routing*- oder *Forward*-Ebene. Hier wird das Paket erneut durchleuchtet.

Bevor es weitergeleitet wird, durchläuft es noch die **output**-Kette, die kontrolliert, ob dieses Paket das System auch verlassen darf.

Jede Kette enthält eine Reihe von Regeln. Jede Regel untersucht den Paketkopf und entscheidet dann über das weitere Verbleiben des Paketes. Dabei wird jedes Paket mit allen Regeln jeder Kette verglichen. Ist die Überprüfung am Ende einer Kette angelangt, entscheidet der Linuxkern, was in der zugehörigen Ketten-Policy steht, und führt die entsprechenden Anweisungen aus.



## IP Chains anwenden

Zur Programmierung des IP Chains-Paketes steht das Kommando **ipchains** zur Verfügung. Mittels dieses Programmes können sowohl neue Ketten als auch Regelsätze definiert und gemanagt werden. Dazu stehen folgende Grundparameter bereit:

Parameter	Bedeutung
-N	neue Kette erzeugen
-X	leere Kette löschen ( <b>input</b> -, <b>forward</b> - und <b>output</b> -Ketten kann man jedoch nicht löschen)
-P	Policy einer Kette ändern
-L	Regelsätze anzeigen
-F	alle Regelsätze einer Kette löschen
-Z	Zähler einer Kette zurücksetzen

Will man eine Regel innerhalb einer Kette ändern, gibt es verschiedene Wege:

Parameter	Bedeutung
-A	neue Regel an eine Kette anfügen
-I	neue Regel in eine Kette einfügen
-R	Regel innerhalb einer Kette ersetzen
-D	Regel innerhalb einer Kette löschen

Diese Befehle sehen wir uns nun anhand von Beispielen genauer an. Das zugrundeliegende Netzwerk sieht so aus:

Auf dem Firewallrechner werden zum Schutz des dahinterliegenden Rechners oder Netzwerkes IP-Chains eingesetzt. Das Firewallsystem verfügt über zwei Netzwerkinterfaces, hier zwei Netzwerkkarten (**eth0** nach außen und **eth1** nach innen). Unser internes Netzwerk hat die Adressen **192.168.2.x**. Um es gegen Angriffe von außen zu schützen, werden Filterregeln definiert. Da ein Schutz gegen Angriffe von außen

```
ipchains -A input -s 0.0.0.0/0.0.0.0 -d 192.168.2.0/255.255.255.0 -j REJECT
```

Dieser Befehl verdeutlicht die Syntax des **ipchains**-Kommandos. Zunächst wird dem Programm durch den Parameter **-A input** mitgeteilt, dass der **input**-Kette eine neue Regel zugefügt wird. Die folgenden Parameter beschreiben die Regel selbst. Das Attribut des Parameters **-s** beschreibt den Adressbereich der Ursprungsadresse, auf den die Regel anzuwenden ist. Dabei wird zunächst die IP-Adresse genannt und dann, durch einen Schrägstrich getrennt, die entsprechende Netzwerkmaske, wobei der Wert **0.0.0.0/0.0.0.0** für alle Adressen gilt. Das Attribut des nächsten Parameters **-d** beschreibt den Zieladressbereich. Dabei entspricht die Notation der des **-s** Parameters. In unserem Beispiel soll diese Regel demnach für Pakete einer beliebigen Ursprungsadresse gelten, welche als Ziel ein System im Netzwerk **192.168.2.x** haben.

Der folgende Parameter **-j** beschreibt die Aktion, die greifen soll, sofern ein Paket diesen Filter erfüllt. Als Attribut wird diesem Parameter ein definiertes Schlüsselwort übergeben. Möglich sind:

aufgebaut werden soll, müssen die von außen kommen Pakete betrachtet werden, also Pakete vom Interface **eth0**.

Sollen alle Pakete von außen zum lokalen Netz abgewiesen werden, benötigt die **input**-Kette eine entsprechende Anweisung, die alle Pakete ins Netzwerk **192.168.2.x** zurückweist.

Dieser Befehl sieht so aus:

<i>Schlüsselwort</i>	<i>Bedeutung</i>
<b>ACCEPT</b>	Paket akzeptieren.
<b>deny</b>	Paket wird als nicht existent angesehen (wird verschluckt).
<b>reject</b>	Paket wird untersagt und der Sender darüber informiert (ICMP unreachable)
<b>redirect</b>	Paket wird an einen lokalen Port weitergeleitet.
<b>masq</b>	Paket wird maskiert (gilt nur in der <b>forward</b> -Kette)

Mit der oben gezeigten Regel werden alle eingehenden Pakete in das lokale Netzwerk zurückgewiesen. Damit ist von außen keinerlei Kommunikation nach innen möglich. Dies ist natürlich nicht unbedingt erwünscht. Denn auch Verbindungen, welche von innen initiiert werden, sind nun nicht mehr möglich, da die Antwortpakete von der Firewall abgewiesen werden.

Wie man sieht, müssen Filterlisten wohl überlegt sein und bedürfen eines gründlichen Konzeptes. Um die Filtermöglichkeiten noch flexibler zu machen, kennt IP-Chains weitere Parameter, die innerhalb der einzelnen Regelsätze greifen:

<i>Parameter</i>	<i>Bedeutung</i>
<b>-p</b> <Protokoll>	spezifiziert ein Protokoll, etwa tcp, udp, icmp, für welches diese Regel Gültigkeit hat.
<b>-y</b>	Prüft das TCP Syn-Bit im TCP-Kopf

Eine weitere Möglichkeit, Filter zu verfeinern, besteht darin, neben IP-Adressbereichen auch Ports anzugeben. Diese werden einfach hinter dem entsprechenden IP-Adressbereich angegeben. Als Trennzeichen wird dabei das Leerzeichen benutzt. So lässt der folgende Filter alle Pakete ins lokale Netzwerk mit Zielport 25 (sprich **smtp**) zu:

```
ipchains -I input -p tcp -s 0.0.0.0/0.0.0.0 -d 192.168.2.0/24 25 -j ACCEPT
```

Eine ausführliche Erläuterung von IP-Chains übersteigt leider deutlich den Umfang dieses Booklets. Wer jedoch Interesse an dieser Technik hat, dem sei das Firewall Handbuch von Guido Stephken nahegelegt, das im Internet unter <http://www.little-idiot.de/firewall> komplett online lesbar ist. Darüberhinaus liefert auch das IPCHAINS-HOWTO sehr gute Informationen – unter folgender Adresse: <http://www.linux-howto.com>

- #) 27  
 (Network Information Center 40  
 .rhosts 19  
 /etc/exports 32  
 /etc/ntp.conf 26  
 /etc/smb.conf 32; 49  
 /etc/smbpasswd 49  
 @-Zeichen 24  
 127.0.0.1 15  
 2<sup>nd</sup> Level 40  
 8-bit-konform 23  
 Adressblock 11  
 Adressfilter 54  
 Adressierungsschema 10  
 Aliases 13  
 anonymous ftp 31  
 Anpassungsschicht 6  
 apachectl 39  
 Apache-Webserver 33  
 Application Gateway 54  
 Arbeitsstation 9  
 arpa 41  
 ASCII 23  
 Base Options 50  
 bin 23  
 Bitmaske 10  
*BOOTP* 13  
 Broadcast-Adresse 11  
 Browse Options 50  
 Cache 40  
 CERT 52  
 Cheapernet 8  
 com 41  
 Config mode 13  
 Daemon 26  
 Datagramm 49  
 Default Gateway 11  
 Default-Route 17  
 Denial of Service 52  
*DHCP* 13  
 directory 42  
 Distribution 13  
 DNS 40  
 DNS-Name 24  
 Domain Name System 40  
 Dos 52  
 drop 17  
 edu 41  
 elm 25  
 Email 46  
 encrypt passwords 51  
 Endwiderstände 7  
 error 17  
*eth0* 14  
 expire 44  
 file 42  
 file' 42  
 Filesystem 25  
 Firewall 25  
 Flags 27  
 forward resolving 41  
 FQDN 13  
 ftp 21  
 FTP-Client 21  
 FTP-Server 30  
 FTP-Shell 21  
 ftp-Sitzung 23  
*Full qualified Domain Name* 13  
 Funktechnik 8  
 Gateway 11; 54  
 Glasfasern 8  
 Globals' 50  
 Grossrechner 5  
 GUI 21  
 Hash 27  
 Header 10  
 hint 42  
 Host name 13  
 Hostname 40  
 hosts allow/hosts deny 51  
 Hubs 7  
 ifconfig 16  
 IMAP 46  
*in-addr* 41  
 INIT-Scripts 52  
 int 41  
 Interfacekonfiguration 16  
 interfaces 50  
 Internet Protocol 10  
 Internet Software Consortium 42  
 IP 6  
 IP Chains 55  
 IP-Adresse 10; 24  
 ipchains 55  
 ipfwadm 55  
 IPX 6; 49  
 IRQ 14  
 ISC 42  
 ISO 5  
 Jokerzeichen 22; 24  
 Kennung 19  
 Koaxialkabel 8  
 Kommandoshell 28  
 Kommandozeilenprogramm 19  
 Kommunikationsprotokoll 8  
 LAN 40  
 Layer 5  
 Lesezugriff 32  
 linuxconf 13  
 LLC 6  
 lo 17  
 Local Area Network 40  
 Logging Options ( 50

- Logical Link Control 6  
Loopback-Adresse 15  
lsmmod 16  
m4 47  
MAC 6  
MAC-Adresse 17  
MacOS 9  
Mailclient 25  
master 42  
Media Access Control 6  
mget 23  
Mikroprozessortechnik 5  
mil 41  
mount 25  
mutt 25  
MX-Record 46  
named 45  
named control 45  
named.conf 42  
named.run 45  
ndc 45  
ne2k-pci 14; 16  
net 41  
Net device 13  
Net News Transport Protocol 25  
NetBEUI 49  
NETBIOS 49  
netbios name 50  
netcfg 13  
*netconfig* 13  
netstat 17  
NetWare 9  
Network File System 24  
Netzmaske 10  
Netzwerkadresse 11  
Netzwerkkartentreiber 16  
Newsserver 25  
NIC 40  
nmbd 49  
nmblookup 49  
nn 26  
NNTP 25  
Novell NetWare 9  
org 41  
OSI 5  
overrun 17  
Paketnummer 10  
Paket-orientiert 10  
Password 19; 51  
pingen 15  
Pointer 43  
POP3 46  
POP-Server 25  
Port 27  
Port 901 50  
Portfilter 54  
Portmapper 32  
Post Office Protocol 25  
Primary name + domain 13  
Printers 51  
proFTPd 30  
proftpd.conf 30  
Protokoll 27  
Proxy 54  
Proxyserver 40  
PTR 45  
Punkt-zu-Punkt-Verbindung 6  
readonly 32  
Rechneradresse 24  
Rechnername 40  
Relay-Server 48  
Remote Copy Programm 24  
Remote Shell 19  
Repeater 8  
Resolving 41  
Resource Record 43  
Ressourcen 19  
Reverse Mapping 43  
reverse resolving 41  
RFC 11  
RG58 8  
rlogin 20  
ro 32  
root 26  
root-Cache 42  
Root-Nameserver 41; 42  
Router 11; 54  
Routing and gateways 14  
Routingprobleme 17  
rsh 19  
r-tools 19  
RX 17  
Samba 25  
Schreibzugriff 32  
security 51  
Security Options 50  
sendmail 47  
*Server* 7; 9  
server string 50  
Serverbetriebssystem 9  
Serviceport 27  
Shares 51  
*Shell* 19  
Simple Mail Transfer Protocol 25  
Simple Mail Transport Protocol 46  
slave 42  
slogin 20  
slrn 26  
smbclient 49  
smbd 49  
smbmount 25  
smbpasswd 49  
SMTP 46  
SMTP-Server 25

SOA 43  
Socket-Type 27  
-Source 33  
SPAM-Relay 48  
squid 40  
squid.conf 40  
ssh 20  
sshd 28  
standalone 31  
Startprogramm 27  
Status 51  
s-Tools 20  
strukturierte Verkabelung 7  
Subdomain 40  
swat 49  
*Switches* 7  
Systemverwalterprivileg 26  
tcpd 53  
TCP-Wrapper 53  
telnetd 28  
Terminalemulation 28  
Terminalsession 19; 28  
testparm 49  
Thin-Ethernet 8  
Timeout 10  
tin 26  
Top Level Domain 40  
Topleveldomains 40  
Topologie 7  
traceroute 18  
Transmission Control Protocol  
10  
TTL 44  
Tuning Options 50  
TX 17  
type 42  
Übertragungsmedium 7  
Umgebungsvariable 28  
Unix 9  
Unixderivat 9  
Userkennung 19  
*Verbindungssegmente* 8  
Verbindungssteuerungsschicht  
6  
Verkabelung, strukturierte 7  
Windows 95/98 9  
Windows NT 9; 49  
Windowsfreigabe 32  
WINS Optionen 50  
workgroup 50  
World Wide Web 33  
YaST 13  
Zeitserver 26  
Zone 42  
Zonendatei 42