X.systems.press

X.systems.press ist eine praxisorientierte Reihe zur Entwicklung und Administration von Betriebssystemen, Netzwerken und Datenbanken.

Rafael Kobylinski

Mac OS X Tiger

Netzwerkgrundlagen, Netzwerkanwendungen, Verzeichnisdienste

Mit 118 Abbildungen und 8 Tabellen

Springer

Dr. Rafael Kobylinski
Manager System Engineers
Apple Deutschland
kobylinski.buch@mac.com

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

ISSN 1611-8618
ISBN-10 3-540-20440-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-20440-4 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Satz und Herstellung: LE-TEX, Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3142 YL – 5 4 3 2 1 0

Inhaltsverzeichnis

1

Einleitung

Mit der Übernahme von Next Ende 1996 begann bei Apple Computer eine neue Ära. Das Ergebnis war die Entwicklung von Mac OS X, einem Betriebssystem mit dem Anspruch, die sprichwörtliche Bedienungsfreundlichkeit eines Apple Macintosh mit dem Leistungsumfang und der Flexibilität einer Unix-Workstation zu vereinigen.

Mac OS X ist eine Weiterentwicklung von OPENSTEP, einer Unix-Variante, deren Entwicklungsgeschichte bei Next bereits in den achtziger Jahren begann. Anders als das ältere, „klassische“ Mac OS unterstützt Mac OS X symmetrisches Multiprocessing (SMP), verfügt über ein modernes virtuelles Speichersystem und beherrscht präemptives Multitasking. In der aktuellen Mac-OS-X-Version 10.4 „Tiger“ besteht der Betriebssystemkern von Mac OS X aus einem Mach 3.0 Mikrokernel und einer auf FreeBSD 5 basierten Betriebssystemschicht. Der Quelltext des Betriebssystemkerns und der Unix-Benutzerumgebung ist frei verfügbar (Projekt Darwin) und als eigenständiges Betriebssystem lauffähig.

Mit derzeit mehr als 15 Millionen Benutzern ist Mac OS X eine der am weitesten verbreiteten Unix-Plattformen überhaupt. Diese große Verbreitung verdankt Mac OS X dabei durchaus auch der Tatsache, dass viele Benutzer das System nicht als eine Unix-Workstation wahrnehmen, sondern als einen einfach zu bedienenden Computer, mit dem sie ihre Büroarbeiten erledigen, ihre Videos schneiden oder ihre Musiksammlung verwalten können. Unter der Bedienungsoberfläche Aqua verbirgt sich jedoch ein voll funktionsfähiges Unix-System (Abb. 1.1).

Mac OS X verfügt Unix-typisch über hervorragende Netzwerkfähigkeiten. Viele dieser Fähigkeiten lassen sich auf Knopfdruck in der graphischen Benutzeroberfläche aktivieren, andere, fortgeschrittene Eigenschaften bedürfen einer manuellen Konfiguration.

Dieses Buch richtet sich an diejenigen, die diese Fähigkeiten erkunden und verstehen wollen. Es beschreibt das Zusammenspiel der graphischen Bedienelemente mit den zugrunde liegenden Systemkomponenten und demons-

Abb. 1.1. Unter der Bedienungsoberfläche Aqua verbirgt sich ein voll funktionsfähiges Unix-System.

triert fortgeschrittene, weit über die Möglichkeiten der graphischen Benutzeroberfläche gehende Konfigurationsmöglichkeiten.

Das Hauptaugenmerk liegt dabei auf der Darstellung der zugrunde liegenden Konzepte und des Zusammenspiels der Systemkomponenten. Die dargestellten Konfigurationsbeispiele zeigen das Potential, haben aber keinen Produktionscharakter, sollten also nicht unreflektiert in eine Unternehmensumgebung übertragen werden.

Insbesondere muss darauf hingewiesen werden, dass dieses Buch nahezu ausschließlich die Einzelplatzversion von Mac OS X behandelt[1] und viele Eigenschaften dieser Version beschreibt, deren korrekte Funktionsweise unter allen erdenklichen Bedingungen weder Apple noch der Autor garantieren können.

Für einen Betrieb außerhalb des Testlabors oder der heimischen Spielwiese sollte man daher zur Serverversion Mac OS X Server greifen, deren Funktionsumfang im Anhang skizziert ist. Mac OS X und Mac OS X Server sind allerdings nur zwei Varianten ein und desselben Betriebssystems, so dass

[1] Inbesondere wird der Name *Server* meistens als Rolle und nicht als Produktbezeichnung gebraucht – ein Rechner mit dem Namen *Tiger-Server.local* übernimmt demnach lediglich die Rolle eines Servers für bestimmte Dienste.

viele der hier dargestellten Eigenschaften der Einzelplatzversion einen ersten Einblick in die Funktionsweise von Mac OS X Server geben.

Die Darstellung richtet sich in jedem Fall nicht an den Anfänger, der gerade die ersten Schritte mit Mac OS X macht und vorher noch nie mit einem Unix-basierten Betriebssystem gearbeitet hat. Stattdessen soll sie dem fortgeschrittenen Benutzer helfen, die Besonderheiten von Mac OS X zu verstehen und einzuordnen.

Sie entspricht dem Stand von Mac OS X 10.4.2 und Xcode 2.1.

1.1 Terminal vs. Finder

Um fortgeschrittene Eigenschaften von Mac OS X zu erkunden ist der Zugriff auf die Unix-Benutzerumgebung unabdingbar. Der Ordner Programme enthält im Unterordner Dienstprogramme die Anwendung Terminal. Terminal startet eine Terminalemulation und eine Shell. In Mac OS X 10.4 ist `bash` als Standard-Shell definiert (Abb. 1.2). Mit dem Befehl `man` kann man sich für nahezu jeden Befehl eine Kurzdokumentation anzeigen lassen.

Mit den üblichen Befehlen wie `pwd`, `ls` und `cd` kann man im Dateisystem navigieren. Dabei fällt auf, dass die Darstellung innerhalb der Terminals von

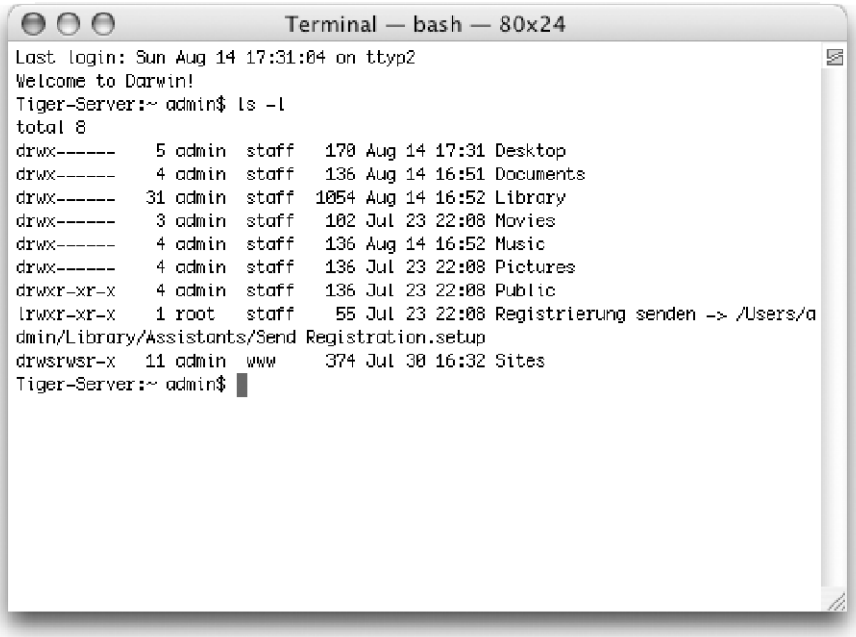


Abb. 1.2. Die Anwendung Terminal erlaubt den Zugriff auf die Unix-Benutzerumgebung.

der Finder-Darstellung abweicht. Im Terminal kommt die tatsächliche Verzeichnisstruktur zum Vorschein, während der Finder mit einer vereinfachten und lokalisierten Darstellung arbeitet.

Der Finder zeigt ähnlich wie `ls` ohne die Option `-a` keine Dateien an, deren Namen mit einem Punkt beginnen. Er respektiert darüber hinaus das HFS-spezifische Attribut *Invisible*, das mit den Befehlen `SetFile` und `GetFileInfo`[2] gesetzt und gelöscht werden kann:

```
$ /Developer/Tools/GetFileInfo /bin
directory: "/bin"
attributes: aVbstclinmedz
created: 03/28/2005 08:42:12
modified: 07/14/2005 22:21:25
```

In älteren Versionen hat der Finder stattdessen die Datei `/.hidden` gelesen, die eine Liste von zu versteckenden Dateien enthielt – in Mac OS X 10.4 wird diese Datei nicht mehr benötigt, der Finder scheint sie aber immer noch zu lesen, falls sie zusätzlich angelegt wird.

Die Boot-Partition bildet die Wurzel (Root `/`) des Verzeichnisbaums. Das im Finder unsichtbare Verzeichnis `/Volumes/` dient als Anknüpfungspunkt für alle zusätzlichen Partitionen oder Datenträger wie etwa zusätzliche interne und externe Festplatten oder Wechselmedien.

Das klassische Mac OS und das Dateisystem HFS unterstützen komplexe Dateien mit einem Data Fork und einem Resource Fork. Solche Dateien sind in Mac OS X zwar sehr selten geworden, aber noch nicht völlig von der Bildfläche verschwunden. Ein prominentes Beispiel sind Finder-Aliase, die sich anders als symbolische Links nicht auf Dateipfade, sondern auf HFS-spezifische File IDs stützen.

Seit Mac OS X 10.4 unterstützten die Befehle zur Manipulation von Dateien wie `cp`, `mv`, `tar` und `rsync` HFS-spezifische Resource Forks. Um explizit auf den Resource Fork zuzugreifen, kann man `/rsrc` an den Dateinamen anhängen (hier ein Finder-Alias):

```
$ ls -l Adressbuch
-rw-r--r--   1 admin  staff  0 Aug 14 18:16 Adressbuch
$ ls -l Adressbuch/rsrc
-rw-r--r--   1 admin  staff  51603 Aug 14 18:16
             Adressbuch/rsrc
```

Dateien kann man mit `rm` und (leere) Verzeichnisse mit `rmdir` entfernen. Falls es mit diesen Befehlen trotz ausreichender Zugriffsrechte nicht klappt, liegt es in der Regel am *Immutable* Flag. Dieses Flag wird vom Finder gesetzt, wenn der Benutzer ein Dokument als *geschützt* kennzeichnet (Abb. 1.3):

```
$ ls -lo Wichtig.txt
-rw-r--r--   1 admin  staff  uchg 0 Aug 14 18:45 Wichtig.txt
```

[2] Diese beiden Befehle werden zusammen mit der Entwicklungsumgebung Xcode installiert.

Abb. 1.3. Für den Schutz von Dokumenten verwendet der Finder das Immutable-Flag.

Im Terminal kann dieses Flag mit dem `chflags`-Befehl entfernt werden:

```
$ chflags nouchg Wichtig.txt

$ ls -lo Wichtig.txt
-rw-r--r--   1 admin  staff  - 0 Aug 14 18:45 Wichtig.txt
```

Für das Editieren von Textdateien steht neben dem graphischen TextEdit eine breite Palette von Terminal-kompatiblen Editoren zur Verfügung. Für den Einsteiger sind die `pico` und `nano` aufgrund der einfachen Bedienung zu empfehlen, für langgediente Unix-Veteranen wird sowohl `emacs` als auch `vi` mitgeliefert.

Der Root-Benutzer (Name `root` und UID 0) existiert, ist aber normalerweise deaktiviert. Empfohlen wird die Verwendung des Befehls `sudo` um temporäre Root-Rechte für die Ausführung einzelner Befehle zu erlangen. Berechtigt sind alle Mitglieder der Gruppe `admin`. In der Systemeinstellung Benutzer kann die Mitgliedschaft in dieser Gruppe über die Checkbox „Dieser Benutzer darf diesen Computer verwalten“ gesteuert werden:

```
$ sudo cat /etc/sudoers
...
# User privilege specification
root    ALL=(ALL) ALL
%admin  ALL=(ALL) ALL
...
```

Entsprechend wird in diesem Buch allen Befehlen, die Root-Rechte erfordern, ein `sudo` vorangestellt. Wem das zu umständlich ist, der kann mit `sudo -s` natürlich auch eine Root-Shell bekommen:

```
$ sudo -s
#
```

Um den Root-Benutzer zu aktivieren (z. B. damit man sich als `root` anmelden kann) muss man ein Root-Passwort setzen. Am einfachsten geht das mit dem Dienstprogramm NetInfo-Manager, das im Menü Sicherheit über einen entsprechenden Befehl verfügt.

1.2 NetInfo und Benutzerkennungen

NetInfo ist ein hierarchisches Datenbanksystem und ein Verzeichnisdienst, dessen Fähigkeiten eingehend im Kapitel 4 dargestellt werden. Andere Unix-Systeme verwenden normalerweise Textdateien, um Konfigurations-, Benutzer- und Gruppeninformationen zu speichern. In Mac OS X werden ausgewählte Konfigurationsdaten sowie Benutzer- und Gruppeneinträge stattdessen in der lokalen NetInfo-Datenbank gespeichert.

Der Inhalt der lokalen NetInfo-Datenbank lässt sich am einfachsten mit dem Dienstprogramm NetInfo-Manager bearbeiten (Abb. 1.4). Innerhalb des NetInfo-Managers kann man Verzeichnisse, Einträge und Attribute anlegen, bearbeiten und löschen. Um Änderungen vorzunehmen ist die Authentifizierung als lokaler Administrator erforderlich. Am eindeutigsten funktionieren dabei die Befehle im Menü Verzeichnisse. Man sollte nur stets darauf achten, dass man zum Bearbeiten oder Löschen im Browser-Fenster immer das richtige Objekt ausgewählt hat.

Für die Auswertung und Bearbeitung von NetInfo-Daten im Terminal stehen die Befehle `nicl`, `nifind`, `nigrep`, `nidump`, `niload` und `niload` zur Verfügung.

Mit `nidump` kann man u. a. eine /etc/passwd-kompatible Liste von Benutzerdaten generieren:

```
$ nidump passwd .
nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false
root:********:0:0::0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false
unknown:*:99:99::0:0:Unknown User:/var/empty:/usr/bin/false
lp:*:26:26::0:0:Printing Services:/var/spool/cups:/usr/bin/false
postfix:*:27:27::0:0:Postfix User:/var/spool/postfix:/usr/bin/false
www:*:70:70::0:0:World Wide Web Server:/Library/WebServer:
    /usr/bin/false
eppc:*:71:71::0:0:Apple Events User:/var/empty:/usr/bin/false
mysql:*:74:74::0:0:MySQL Server:/var/empty:/usr/bin/false
sshd:*:75:75::0:0:sshd Privilege separation:/var/empty:
     /usr/bin/false
qtss:*:76:76::0:0:QuickTime Streaming Server:/var/empty:
     /usr/bin/false
```

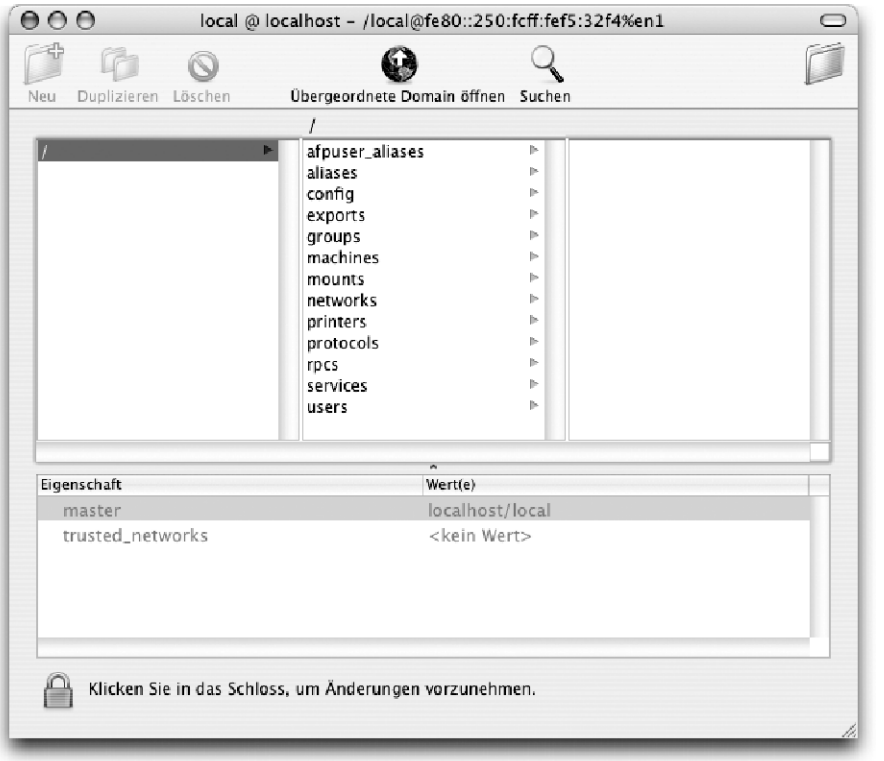


Abb. 1.4. Mit dem NetInfo-Manager kann man den Inhalt der lokalen NetInfo-Datenbank sichtbar machen und bearbeiten.

```
cyrusimap:*:77:6::0:0:Cyrus IMAP User:/var/imap:/usr/bin/false
mailman:*:78:78::0:0:Mailman user:/var/empty:/usr/bin/false
appserver:*:79:79::0:0:Application Server:/var/empty:/usr/bin/false
clamav:*:82:82::0:0:Clamav User:/var/virusmails:/bin/tcsh
amavisd:*:83:83::0:0:Amavisd User:/var/virusmails:/bin/tcsh
jabber:*:84:84::0:0:Jabber User:/var/empty:/usr/bin/false
xgridcontroller:*:85:85::0:0:Xgrid Controller:
                /var/xgrid/controller:/usr/bin/false
xgridagent:*:86:86::0:0:Xgrid Agent:/var/xgrid/agent:/usr/bin/false
appowner:*:87:87::0:0:Application Owner:/var/empty:/usr/bin/false
windowserver:*:88:88::0:0:WindowServer:/var/empty:/usr/bin/false
tokend:*:91:91::0:0:Token Daemon:/var/empty:/usr/bin/false
securityagent:*:92:92::0:0:SecurityAgent:/var/empty:/usr/bin/false
admin:********:501:20::0:0:Administrator:/Users/admin:/bin/bash
```

Diese Liste enthält keine Crypt-Hashes. Im Auslieferungszustand enthalten die Benutzereinträge in NetInfo das Attribut `authentication_authority` mit dem Wert `;ShadowHash;`:

```
$ nicl . -read /users/rkk authentication_authority
authentication_authority: ;ShadowHash;
```

Dadurch wird veranlasst, dass die Hashes lokaler Passwörter in einem nur für Root lesbaren Verzeichnis abgelegt werden:

```
$ sudo ls -1 /var/db/shadow/hash
D645CC33-3BD9-4A8A-8C92-66EDC0063786
D645CC33-3BD9-4A8A-8C92-66EDC0063786.state
```

Die Dateien in diesem Verzeichnis werden den Benutzern mit Hilfe eines eindeutigen Identifikators zugeordnet. Jeder Benutzereintrag enthält ein Attribut `generateduid`, welches diesen Identifikator enthält.

```
$ nicl . -read /users/admin generateduid
generateduid: D645CC33-3BD9-4A8A-8C92-66EDC0063786
```

Zu jedem Benutzer gehören zwei Dateien. Die Datei mit der Endung `.state` enthält dann einige Verwaltungsinformationen:

```
$ sudo cat \
> /var/db/shadow/hash/
  D645CC33-3BD9-4A8A-8C92-66EDC0063786.state
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0
//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CreationDate</key>
        <date>2005-07-23T20:08:48Z</date>
        <key>FailedLoginCount</key>
        <integer>0</integer>
        <key>LastLoginDate</key>
        <date>2005-08-14T17:49:34Z</date>
        <key>NewPasswordRequired</key>
        <integer>0</integer>
</dict>
</plist>
```

Die Datei ohne Endung enthält einen oder mehrere Passwort-Hashes. Mac OS X selbst verwendet einen Salted SHA1 Hash. Sofern Windows Filesharing aktiviert wurde (und nur dann), wird für ausgewählte Benutzer darüber hinaus auch ein Windows-NT Hash (für Windows NT/XP) und ein Lan-Manager Hash (für ältere Windows-Versionen) gespeichert (Abb. 1.5).

Das Attribut `authentication_authority` wird in diesem Fall automatisch um die Liste der zu speichernden Hashes ergänzt:

```
$ nicl . -read /users/admin authentication_authority
authentication_authority:
;ShadowHash;HASHLIST:<SALTED-SHA1,SMB-NT,SMB-LAN-MANAGER>
```

Lan-Manager Hashes gelten heute als unsicher. Sofern man keine älteren Windows-Maschinen einsetzt, kann man die Speicherung dieser Hashes unterdrücken, indem man das Kennwort `SMB-LAN-MANAGER` aus dem Attribut `authentication_authority` entfernt:

```
$ nicl . -read /users/admin authentication_authority
authentication_authority:
;ShadowHash;HASHLIST:<SALTED-SHA1,SMB-NT>}
```

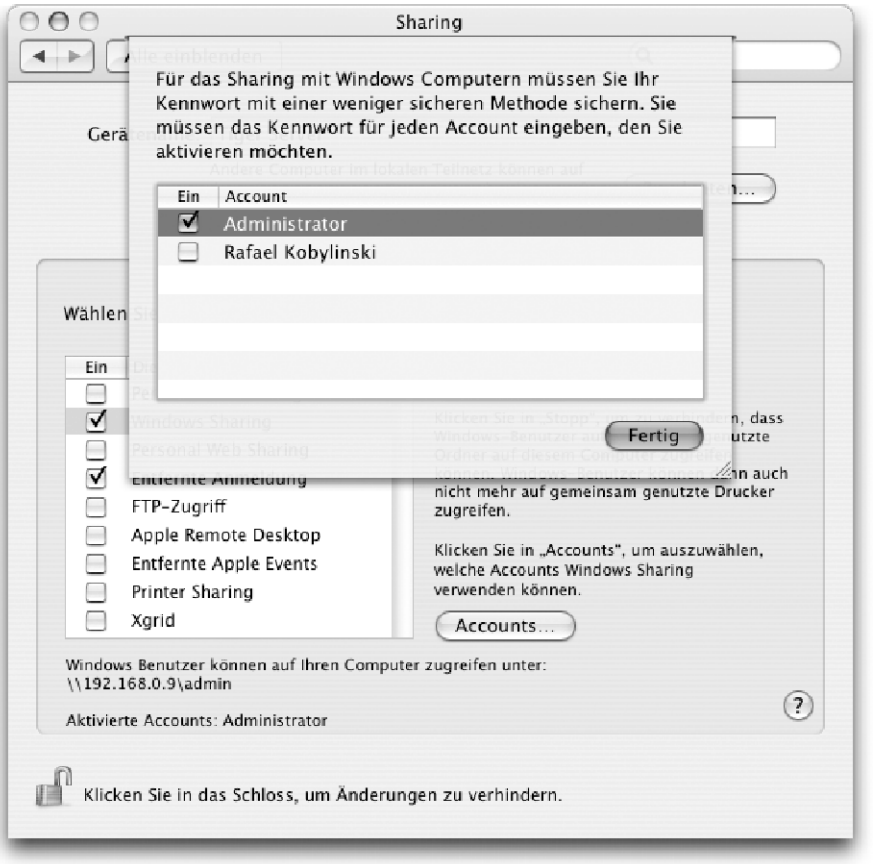


Abb. 1.5. Windows Sharing erfordert die Speicherung von zusätzlichen Hashes.

Um einen bereits gespeicherten Hash zu löschen, muss man das Passwort zurücksetzen oder den gespeicherten Hash einfach mit Nullen überschreiben.

1.3 Drucken mit CUPS

Die Druckerunterstützung von Mac OS X basiert auf dem *Common UNIX Printing System* (kurz CUPS). Mit dem Drucker-Dienstprogramm verfügt Mac OS X über eine graphische Schnittstelle (Abb. 1.6), mit der man komfortabel Drucker einrichten, konfigurieren und überwachen kann.

Darüber hinaus verfügt CUPS über eine eingebaute Web-Schnittstelle, mit der man u. a. auch auf die vollständige Dokumentation dieses Systems zugreifen kann (Abb. 1.7). Der Zugriff auf die Web-Schnittstelle ist lokal über den Port 631 möglich:

```
http://localhost:631
```

Abb. 1.6. Mit dem Drucker-Dienstprogramm kann man Drucker einrichten, konfigurieren und überwachen.

Um von der Kommandozeile aus zu drucken, kann man den Befehl `lp` oder den Befehl `lpr` verwenden. Um eine Liste der verfügbaren Drucker auszugeben verwendet man den Befehl `lpstat`:

```
$ lpstat -p -d
printer HP_LaserJet_2200__0001E69CE741_ is idle.
enabled since Jan 01 00:00
system default destination: HP_LaserJet_2200__0001E69CE741_
```

Die Konfigurationsdaten von CUPS befinden sich sich in `/etc/cups/`. Dort findet man unter anderem auch eine Liste der konfigurierten Drucker und die für diese Drucker benötigten PPD-Dateien:

```
 sudo cat /etc/cups/printers.conf
# Printer configuration file for CUPS v1.1.23
# Written by cupsd on Mon Aug 15 15:18:27 2005
<DefaultPrinter HP_LaserJet_2200__0001E69CE741_>
Info HP LaserJet 2200
DeviceURI mdns://HP%20LaserJet%202200%20%280001E69CE741%29
                ._pdl-datastream._tcp.local./?bidi
State Idle
Accepting Yes
Shared Yes
JobSheets none none
QuotaPeriod 0
```

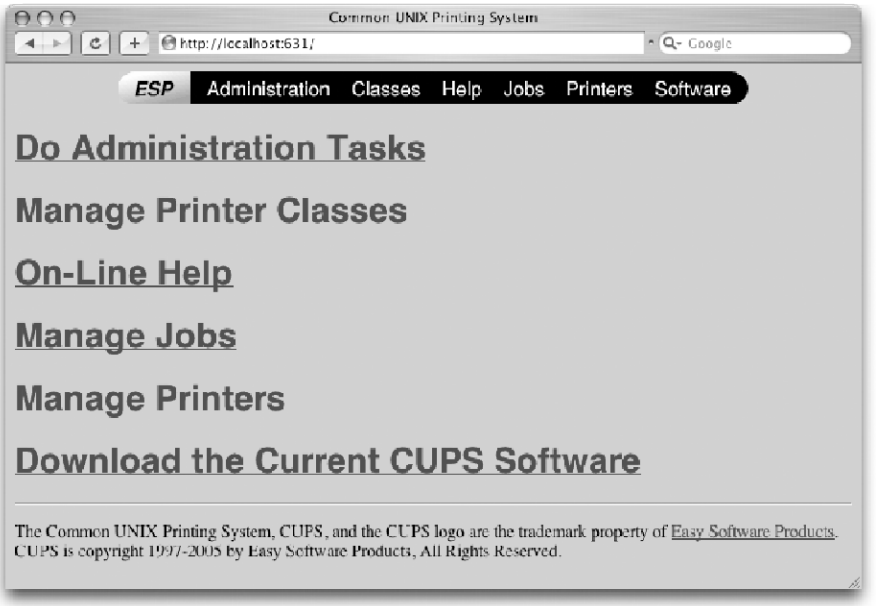


Abb. 1.7. Über die eingebaute Web-Schnittstelle kann man auf die vollständige Dokumentation von CUPS zugreifen.

```
PageLimit 0
KLimit 0
</Printer>

$ ls /etc/cups/ppd/
HP_LaserJet_2200__0001E69CE741_.ppd
```

Um auf einem bestimmten Drucker zu drucken muss man den Druckernamen angeben. Der Befehl `lp` verwendet hierfür die Option `-d` und der Befehl `lpr` die Option `-P`. Druckoptionen wie Papierformat, Ausrichtung oder Größe können über die Option `-o` definiert werden.

Eine Liste der aktiven Druckjobs erhält man mit `lpstat`. Um einen Druckjob zu löschen, kann man entweder `cancel` oder `lprm` verwenden. Details findet man im CUPS Software Users Manual:

```
http://localhost:631/sum.html
```

Als Eingabe werden neben PostScript und PDF auch viele andere Formate unterstützt, z. B. GIF, PNG, JPEG und TIFF. Nicht unterstützte Formate müssen vor dem Drucken in eines der unterstützten umgewandelt werden.

Um beispielsweise eine man-Page mit Hervorhebungen zu drucken, muss man sie zuerst in ein druckbares Format umwandeln. Um den Quelltext einer man-Page in PostScript zu konvertieren, kann man den Befehl `groff` verwenden. Mit Hilfe einer Pipe kann man das Ergebnis direkt zum Drucken an den Standarddrucker weiterleiten:

```
$ groff -Tps -man /usr/share/man/man8/bootpd.8 | lpr
```

Lange man-Pages kann man natürlich auch doppelseitig drucken, sofern man über einen Drucker mit einer Duplex-Option verfügt:

```
$ groff -Tps -man /usr/share/man/man8/bootpd.8 | \
> lpr -o sides=two-sided-long-edge
```

Die Quelltexte der mitgelieferten man-Pages befinden sich im Verzeichnis `/usr/share/man/` und sind kapitelweise geordnet. Der Quelltext der man-Page des Befehls `bootpd` befindet sich beispielsweise im achten Kapitel. Die Zuordnung der man-Pages zu Kapiteln steht normalerweise gleich in der ersten Zeile der man-Page als Zahl in Klammern:

```
BOOTPD(8)       BSD System Manager's Manual  0     BOOTPD(8)

NAME
     bootpd -- DHCP/BOOTP/NetBoot server

SYNOPSIS
     bootpd [options]

DESCRIPTION
     bootpd implements a DHCP/BOOTP server as defined in
     RFC951, RFC1542, RFC2131, and RFC2132.  It is also
     a NetBoot server implementing Apple- proprietary
     NetBoot 1.0 (BOOTP-based) and NetBoot 2.0 BSDP (Boot
     Server Discovery Protocol).  BSDP works along with
     regular DHCP, using DHCP-format packets with a special
     vendor-class identifier and vendor-specific options.
...
```

1.4 Programmvoreinstellungen

Mac OS X verfügt über ein einheitliches System zur Handhabung von Programmvoreinstellungen. Nahezu alle Mac-OS-X-Anwendungen – unabhängig davon, ob sie von Apple selbst oder von Drittherstellern stammen – verwenden dieses System. Ausnahmen bilden vor allem Open-Source-Anwendungen, die möglichst unverändert auf einer Vielzahl von Plattformen laufen müssen. Diese speichern ihre Voreinstellungen häufig wie auf anderen Plattformen in /etc/ und verwenden ein eigenes Datenformat. Prominente (mitgelieferte) Beispiele sind z. B. Apache (mit Voreinstellungen in /etc/httpd/httpd.conf), Samba (mit Voreinstellungen in /etc/smb.conf) und OpenLDAP (mit Voreinstellungen in /etc/openldap/slapd.conf).

In der Regel stellen Mac-OS-X-Anwendungen eine graphische Benutzeroberfläche zur Verfügung, mit der sich die Voreinstellungen bequem und einfach editieren lassen. Manchmal fehlt diese Möglichkeit jedoch komplett oder beschränkt sich nur auf einen Teil der tatsächlich implementierten Parameter. In diesen Fällen lassen sich die gewünschten Programmvoreinstellungen dann u. U. nur per Hand direkt manipulieren.

Systemweite Voreinstellungen werden grundsätzlich im (nur für Administratoren schreibbaren) Verzeichnis /Library/Preferences/ und benutzerspezifische Voreinstellungen im Verzeichnis ~/Library/Preferences/ des betreffenden Benutzers abgelegt. Voreinstellungen, die potentiell für alle Anwendungen gelten können, werden dabei in beiden Fällen in einer Datei mit dem Namen .GlobalPreferences.plist gespeichert. Da dieser Dateiname mit einem Punkt beginnt, ist diese Datei im Finder normalerweise unsichtbar.

Zusätzlich kann die Gültigkeit von Voreinstellungsdateien auf einen bestimmten Host beschränkt werden. Hostspezifische Voreinstellungen werden dazu im Unterverzeichnis ByHost abgelegt und durch die eindeutige MAC-Adresse des Hosts gekennzeichnet.

Anwendungsentwickler werden von Apple ermutigt, ihre anwendungsspezifischen Voreinstellungsdateien nach einem festen Schema zu benennen, um Namenskollisionen zu vermeiden. Als Name der Voreinstellungsdatei soll der eindeutige Anwendungsidentifikator (engl. Application ID) dienen, der aus dem eindeutigen DNS-Namen des entwickelnden Unternehmens und der Anwendungsbezeichnung abzuleiten ist. Geschrieben wird das Ganze dann in der umgekehrten Reihenfolge, also mit der TLD (top level domain) zuerst. Der Anwendungsidentifikator des Mac OS X Finders lautet demnach z. B. com.apple.Finder. Auf diese Weise ist der Entwickler selbst dafür verantwortlich, die Eindeutigkeit der Namen innerhalb der Produktlinie seines Unternehmens zu gewährleisten. Namenskollisionen werden vermieden, da DNS-Namen eindeutig zugeordnet werden können.

Das innerhalb von Voreinstellungsdateien mit Abstand am häufigsten verwendete Format ist das von Apple favorisierte Property List Format (Endung .plist). Das Property List Format erlaubt die Speicherung von einfachen Datentypen in einer einheitlichen und von der verwendeten Rechnerarchitektur unabhängigen Art und Weise.

Innerhalb einer Property List sind einfache und zusammengesetzte Datentypen erlaubt. Als einfache Datentypen werden Zeichenketten (Strings), Gleitkommazahlen, ganze Zahlen, Datumsangaben, Boole'sche Wahrheitswerte sowie Binärdaten[3] unterstützt. Als zusammengesetzte Datentypen können geordnete Reihungen (Arrays) und Hash-Tabellen (Dictionaries) verwendet werden, um aus den einfachen Grundelementen komplexere Strukturen zu erstellen.

Mac OS X 10.4 kennt drei Varianten des Property List Formats: die Binärvariante, die XML-Variante und die ASCII-Variante. Die Binärvariante ist für Menschen nicht lesbar, kann aber problemlos in die lesbare XML-Variante konvertiert werden:

```
$ plutil -convert xml1 Test.plist
```

[3] Das Property List Format ist für die Speicherung von Zeichenketten optimiert. Die Speicherung großer Mengen von Binärdaten ist nicht üblich und auch nicht besonders effizient.

Umgekehrt lässt sich natürlich die XML-Variante in die Binärvariante konvertieren:

```
$ plutil -convert binary1 Test.plist
```

Die XML-Variante des Property List Formats lässt sich relativ leicht lesen. Die gespeicherten Datentypen werden einfach in das XML-Element `<plist>` eingebettet:

```
$ cat XML.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0
//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
...
</plist>
```

Die ASCII-Variante lässt sich ebenfalls leicht lesen, lässt eine Kennzeichnung von Datentypen aber nicht zu. Alle Datentypen werden wie Zeichenketten behandelt:

```
$ cat ASCII.plist
{
        foo = 123;
}
```

Für einzelne Voreinstellungen wird innerhalb einer Property List normalerweise eine Hash-Tabelle (XML-Element `<dict>`) angelegt. Die individuellen Voreinstellungswerte werden dann innerhalb der Hash-Tabelle mit einem Schlüsselwort (XML-Element `<key>`) versehen und in einem zum jeweiligen Datentyp passenden XML-Element (Tabelle 1.1) abgelegt.

Eine einfache Property List mit einer einzigen (fiktiven) Voreinstellung `foo` und dem Wert `123` hätte demnach das folgende Aussehen:

```
$ cat Test.plist <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0
//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
```

Tabelle 1.1. Elemente einer Property List

Datentyp	XML-Element
Zeichenkette	`<string>`
Gleitkommazahl	`<real>`
Ganze Zahl	`<integer>`
Datum	`<date>`
Boole'scher Wahrheitswert	`<true/>` oder `<false>`
Kodierte Binärdaten	`<data>`
Geordnete Reihung	`<array>`
Hash-Tabelle	`<dictionary>`

```
<plist version="1.0">
<dict>
        <key>foo</key>
        <integer>123</integer>
</dict>
</plist>
```

Die XML-Variante des Property List Formats lässt sich natürlich mit einem beliebigen Texteditor problemlos editieren. Einfacher geht es allerdings mit dem Property List Editor, der zum Lieferumfang der Mac-OS-X-Entwicklungsumgebung Xcode gehört und alle Varianten des Property List Formats (Binär, XML und ASCII) unterstützt. Das Installationspaket für Xcode ist auf aktuellen Macs vorinstalliert und muss nur noch installiert werden. Besitzer älterer Geräte können Xcode kostenlos bei Apple (http://developer.apple.com) herunterladen.

Nach der Installation von Xcode findet man den Property List Editor im Verzeichnis /Developer/Applications/Utilities/. Ist das Progamm installiert, lassen sich Dateien mit der Endung .plist komfortabel per Doppelklick öffnen. Der Property List Editor kann dabei sowohl mit der XML-Variante als auch mit der Binärvariante des Property List Formats umgehen.

Einzelne Attribute lassen sich mit dem Befehl `defaults` editieren. Um ein Attribut zu schreiben, muss man entweder den vollständigen Pfad der

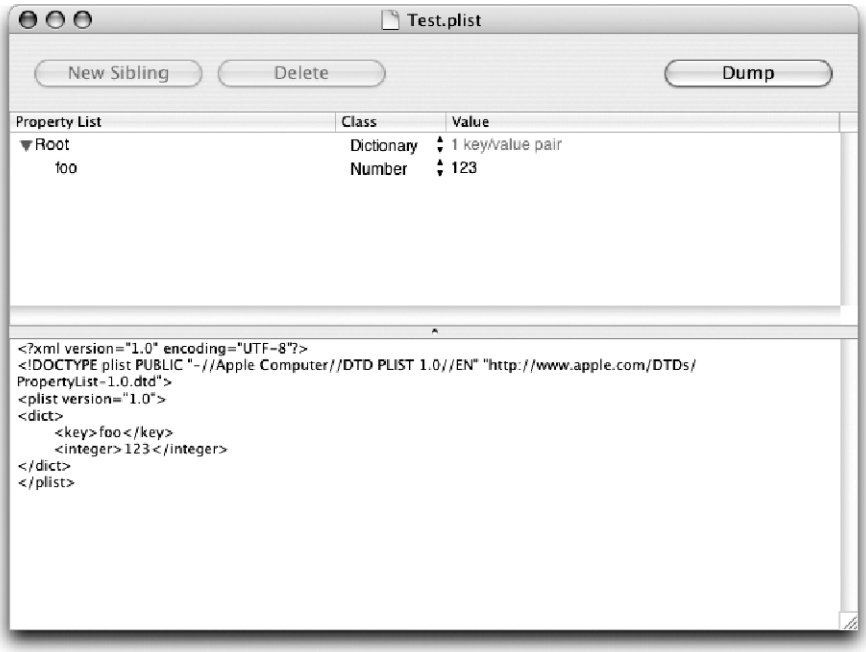


Abb. 1.8. Der Property List Editor unterstützt alle Varianten des Property List Formats.

Zieldatei (ohne die Endung `.plist`) oder die so genannten Domäne angeben. Jeder Anwendung entspricht dabei eine eigene Domäne, die mit dem eindeutigen Anwendungsidentifikator bezeichnet wird. Die globale Domäne `NSGlobalDomain` ist für anwendungsübergreifende Voreinstellungen gedacht.

Ist die Zieldatei noch nicht existent, wird sie von `defaults` angelegt. Dateien im XML- und im ASCII-Format werden automatisch in das binäre Format konvertiert:

```
$ defaults write ~/Desktop/Test Crash -bool NO
```

Um den Inhalt der Datei `Test.plist` nach einer solchen Operation zu betrachten, muss man sie zuerst in das XML-Format konvertieren:

```
$ plutil -convert xml1 Test.plist
```

Der Erfolg der Schreiboperation kann dann auf einfache Weise verifiziert werden:

```
$ cat Test.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0
//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Crash</key>
        <false/>
</dict>
</plist>
```

Umgekehrt kann man mit `defaults` beliebige Attribute auslesen. Die Ausgabe erfolgt dabei im ASCII-Format:

```
$ defaults read ~/Desktop/Test
{Crash = 0; }
```

Diese Ausgabe lässt sich in einer Datei speichern und mit `defaults` bearbeiten. Bei Schreiboperationen wird das ASCII-Format dann wieder in das binäre Format konvertiert.

Mit `defaults` kann man u. a. die Umask ändern. Normalerweise verwendet Mac OS X die Umask 0022:

```
$ umask
0022
```

Um die Umask für die Kommandozeile zu ändern, muss man lediglich eine entsprechende `umask`-Anweisung in eine der globalen oder benutzerspezifischen Shell-Initialisierungsdateien einfügen:

```
/etc/profile
/etc/bashrc
~/.profile
~/.bashrc
```

Diese Einstellungen werden von GUI-Programmen jedoch ignoriert. Mit der offiziell nicht unterstützten Voreinstellung `NSUmask` kann man jedoch auch die Umask der GUI-Programme beeinflussen[4]:

```
$ defaults write -g NSUmask 2
```

Alle solchen Änderungen sind natürlich mit Vorsicht zu genießen. Keiner kann garantieren, dass sie unter allen Bedingungen und frei von Seiteneffekten funktionieren. Eine im Vergleich zum Auslieferungszustand restriktivere Umask kann z. B. zu Problemen bei Softwareinstallationen führen.

[4] Der Wert von NSUmask ist eine Zahl in Dezimalschreibweise. Der (oktale) Wert der Standard-Umask 0022 entspricht z. B. dem dezimalen Wert 18.

2

Netzwerk – Grundlagen

Computer werden heutzutage nur noch in den allerseltensten Fällen im reinen Einzelplatzbetrieb verwendet. Stattdessen wird die Mehrzahl der heute betriebenen Computer bzw. Rechner (zumindest zeitweise) in ein Rechnernetz eingebunden. Diesen Anforderungen entsprechend sind heutige Computer bereits im Auslieferungszustand sowohl hardware- als auch softwaremäßig in der Lage, mit anderen Computern zu kommunizieren.

Mit Hilfe eines Rechnernetzes lassen sich Kostenvorteile erzielen, weil beispielsweise verhältnismäßig teure Peripheriegeräte wie Laserdrucker u.ä. von mehreren Benutzern gleichzeitig benutzt werden können. Damit entfällt die Notwendigkeit, teure Peripherie für jeden Arbeitsplatz einzeln anzuschaffen.

Rechnernetze ermöglichen eine effiziente gemeinsame Nutzung von Datenbeständen, indem lokale Verzeichnisse einfach für die Nutzung im Netzwerk freigegeben werden – das mühsame Kopieren mit Hilfe von physischen Datenträgern entfällt.

Sie bilden schließlich überhaupt erst die Voraussetzung für viele nützliche Kommunkationsanwendungen, die heutzutage als selbstverständlich vorausgesetzt werden: Email, World-Wide Web, Instant Messaging und vieles mehr.

Die Grundlage für alle diese Anwendungen bildet die (primitive) Fähigkeit, einzelne Bits oder auch Folgen von Bits zwischen zwei oder mehr Rechnern auszutauschen. Dazu bedarf es eines gemeinsamen Übertragungsmediums, Hardwarekomponenten, welche die Rechner mit dem Übertragungsmedium verbinden, sowie festgelegter Verfahren für die Nutzung des Übertragungsmediums.

Standards legen dabei sowohl die physischen Eigenschaften des zu verwendenden Übertragungsmediums als auch die Verfahren zur seiner Nutzung fest. Nur auf der Grundlage von Standards lassen sich Rechnernetze aus Komponenten unterschiedlicher Hersteller aufbauen und betreiben.

Zu den heutzutage gängigsten Standards gehören die beiden IEEE Standards 802.3 und 802.11. Diese beiden Standards definieren die Verfahren für die Übertragung von Daten in lokalen Rechnernetzen. Während der ers-

te (IEEE Standard 802.3) die physischen Eigenschaften und die Verfahren für kabelgebundene Rechnernetze festlegt, sind im zweiten (IEEE Standard 802.11) analoge Festlegungen für drahtlose Netzwerke getroffen worden.

Unabhängig vom jeweiligen physischen Übertragungsmedium werden alle mit IEEE 802.3 oder IEEE 802.11 konformen Netzwerke als *Ethernets*[1] bezeichnet, weil sie auf dem Prinzip eines von allen angeschlossenen Rechnern geteilten Übertragungsmediums beruhen. Dieses Prinzip und die für das jeweilige Übertragungsmedium maßgeblichen physikalischen Gesetze setzen sowohl der maximal möglichen Übertragungsgeschwindigkeit als auch der maximalen Netzwerkgröße enge Grenzen.

Um weit voneinander entfernte Rechner zu verbinden, sind demnach regelmäßig andere Technologien notwendig als für die Verbindung von Rechnern innerhalb eines Gebäudes oder eines Stockwerks: Man unterscheidet entsprechend LAN- (Local Area Network) und WAN-Technologien (Wide Area Network), wobei Ethernet in allen existierenden Ausprägungen eindeutig dem LAN-Bereich zuzuordnen wäre und das heutzutage für die Anbindung an das weltweite Internet häufig verwendete ADSL (Asynchronous Digital Subscriber Line) ein Beispiel für eine Technologie aus dem WAN-Bereich darstellt.

Um den Anwender der Komplexität all dieser Übertragungsverfahren nicht aussetzen zu müssen, wurden zusätzliche Verfahren – Netzwerkprotokolle – entwickelt, welche die Kommunikation von Rechner zu Rechner in Unabhängigkeit von der die beiden Kommunikationspartner verbindenden Übertragungstechnologie ermöglichen. Um ein Übertragungsmedium mit einem solchen Netzwerkprotokoll nutzen zu können, bedarf es lediglich eines Adapters, welcher der Protokollsoftware den Zugriff auf das Medium ermöglicht.

2.1 Die TCP/IP-Protokollfamilie

Die Entwicklung der TCP/IP-Protokollfamilie begann in den sechziger Jahren im Rahmen eines Forschungsprojekts der amerikanischen Regierung. Zu jener Zeit waren Rechnernetze etwas, was lediglich wenige Spezialisten in Forschungseinrichtungen, in Universitäten und beim Militär beschäftigte.

In den neunziger Jahren wurden Rechnernetze jedoch zunehmend zu einem Massenphänomen und eroberten nicht nur die meisten Unternehmen, sondern auch viele Privathaushalte. Nicht zuletzt deswegen, weil TCP/IP die Grundlage für den weltweit größten Netzwerkverbund – das *Internet* – bildet, hat sich diese Protokollfamilie gegenüber konkurrierenden Technologien durchgesetzt.

Heutzutage is TCP/IP der Standard für die Rechnervernetzung. Während im klassischen Mac OS noch die proprietäre AppleTalk-Protokollfamilie do-

[1] Abgeleitetvon Äther, dem vermeintlichen Ausbreitungsmedium für elektromagnetische Wellen, an dessen Existenz Physiker bis Ende des 19 Jh. glaubten.

minierte, basieren in Mac OS X nun nahezu alle Netzwerkdienste und -werkzeuge auf TCP/IP.

Die Netzwerkfähigkeiten von Mac OS X basieren unmittelbar auf der 4.4BSD TCP/IP Implementierung. Diese Referenzimplementierung findet man außer in Mac OS X auch in anderen BSD basierten Betriebssystemen wie FreeBSD, OpenBSD oder NetBSD. Auf dieser Grundlage können Mac-OS-X-Rechner nicht nur sehr einfach in bestehende TCP/IP Netzwerke als *Clients* eingebunden werden, sondern können problemlos auch nahezu alle in solchen Netzwerken üblichen Dienste als *Server* bereitstellen.

2.1.1 Das TCP/IP-Schichtenmodell

Netzwerkprotokolle lassen sich in der Regel sehr gut mit Hilfe eines Schichtenmodells beschreiben. Für das Verständnis von TCP/IP genügen dabei fünf Schichten: die physikalische Schicht, die Verbindungsschicht, die Netzwerkschicht, die Transportschicht und die Anwendungsschicht.

1. Die *physikalische Schicht* verbindet die Kommunikationspartner mit Hilfe eines Übertragungsmediums, z. B. eines Kabels, mit genau definierten technischen Eigenschaften.
2. Die *Verbindungsschicht* stellt die grundsätzliche Fähigkeit her, Datenpakete über ein physisches Medium, wie beispielsweise ein Ethernet-Kabel oder über Funk, von einem Rechner zum anderen zu versenden.
3. Die *Netzwerkschicht* nutzt die Verbindungsschicht, um den Versand von Datenpaketen auch dann zu ermöglichen, wenn die beteiligten Rechner nicht unmittelbar durch ein physisches Medium verbunden sind. Ein wesentlicher Bestandteil der Netzwerkschicht von TCP/IP ist IP, das *Internet-Protocol*. Die Hauptaufgabe von IP ist entsprechend die Bereitstellung von Mechanismen zur Vermittlung von Datenpaketen über unterschiedliche Rechnernetze hinweg (daher auch der Name *Inter*-Net).
4. Die *Transportschicht* nutzt wiederum die Netzwerkschicht, um Anwendungen den Versand von (Nutz-)Daten zu ermöglichen. Dazu stellt die TCP/IP-Protokollfamilie zwei Protokolle zur Verfügung: TCP (*Transmission Control Protocol*) und UDP (*User Datagram Protocol*).
   TCP eröffnet einen verlässlichen Datenkanal zwischen zwei Anwendungen, über den Daten zuverlässig in beide Richtungen fließen können. TCP teilt dabei den Datenstrom in einzelne, mit der Netzwerkschicht kompatible Datenpakete und kümmert sich um deren sichere Zustellung in der richtigen Reihenfolge.
   Das im Vergleich zu TCP viel einfachere UDP ermöglicht lediglich den Versand von einzelnen Datenpaketen (den so genannten *Datagrammen*). Die Anwendungen müssen sich um die korrekte Reihenfolge und die Behandlung von verloren gegangenen Paketen selbst kümmern.
5. Die *Anwendungsschicht* besteht schließlich aus all jenen Anwendungsprogrammen, welche die Kommunikationsdienste der Transportschicht be-

nutzen. Im Kontext von Mac OS X sind damit sowohl alle schwergewichtigen Anwendungsprogramme mit Netzwerkfunktionen wie beispielsweise Web-Browser (wie z. B. Safari oder Internet Explorer) und EMail Clients (wie z. B. Apple Mail), als auch all die vielen kleinen netzwerkbezogenen Befehle, die in der Kommandozeile aufgerufen werden können (wie z. B. telnet, ftp oder ssh), gemeint.

2.1.2 Internet-Adressen

Ein TCP/IP-Rechnernetz besteht aus einer Menge von Rechnern, den so genannten Hosts. Jeder dieser Hosts muss mindestens eine Netzwerkschnittstelle besitzen, wobei dieser Schnittstelle eine Internet-Adresse – kurz IP-Adresse – zugewiesen sein muss.

Bei einer IP-Adresse handelt es sich um eine 32 Bit bzw. 4 Bytes lange Zahl, welche die Schnittstelle eindeutig bezeichnet. Bei der Darstellung von IP-Adressen ist es üblich, die vier Bytes hintereinander, jeweils durch einen Punkt getrennt, in dezimaler Form zu schreiben: z. B. 17.112.152.32 (Binär 00010001 01110000 10011000 00100000):

	3	2	1	0
Binärdarstellung	00010001	01110000	10011000	00100000
Dezimaldarstellung	17	112	152	32

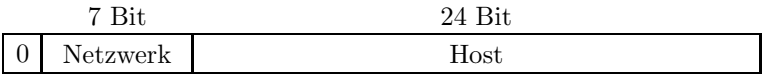
Ein TCP/IP-Rechnernetzverbund besteht aus einer Menge von TCP/IP-Rechnernetzen, die durch Vermittlungsstationen, so genannte Gateways bzw. Router, verbunden sind. Router haben mindestens zwei Netzwerkschnittstellen mit unterschiedlichen IP-Adressen. Sie sind in der Lage, IP-Datenpakete, die sie an einer Netzwerkschnittstelle empfangen, über eine der anderen Netzwerkschnittstellen weiterzuleiten.

Ein Host, der ein IP-Datenpaket verschicken möchte, muss in einem Rechnernetzverbund entscheiden können, ob er den Zielrechner direkt oder nur über einen Router erreichen kann. Dazu sind in jeder IP-Adresse eine Netzwerkadresse und eine Hostadresse codiert. Hosts, die sich in ein und demselben Rechnernetz befinden, haben die gleiche Netzwerkadresse, aber unterschiedliche Hostadressen.

Der sendende Host vergleicht vor dem Versenden eines IP-Pakets die Zieladresse mit der IP-Adresse seiner eigenen Netzwerkschnittstelle und kann so feststellen, ob die Netzwerkadressen übereinstimmen oder nicht, und sich auf diese Weise eindeutig für oder gegen die direkte Zustellung entscheiden.

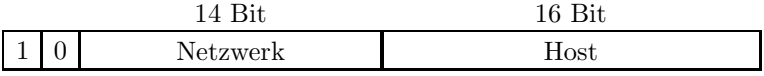
Die Einteilung in Netzwerkadresse und Hostadresse richtete sich in der Vergangenheit ausschließlich nach der so genannten Klasse der IP-Adresse:

- Die *Klasse A* ist für sehr große Netzwerke vorgesehen. Das höchste Bit ist bei solchen Adressen immer 0, die verbleibenden 31 Bits legen das Netzwerk (7 Bits) und die Hostadresse (24 Bits) fest:



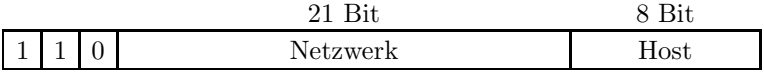
In der üblichen Dezimal-Notation kann also das erste Byte als die Netzwerkadresse (0–127) und die restlichen drei Bytes als die Hostadresse interpretiert werden. Damit gibt es maximal $2^7 = 128$ solcher Klasse-A-Netzwerke, mit jeweils $2^{24} - 2 = 16.777.214$ (die erste und die letzte Hostadresse haben eine spezielle Bedeutung und können nicht für Hosts verwendet werden) Hostadressen. Adressen dieses Typs wurden Anfang der neunziger Jahre vor allem amerikanischen Großunternehmen sowie Forschungs- und Militäreinrichtungen zugewiesen. Apple wurde z. B. das A-Klasse-Netzwerk mit der Netzwerkadresse 17 zugeteilt. Die Netzwerke 0 und 127 haben eine Sonderstellung: das Netzwerk 0 steht für die so genannte *Default Route* und das Netzwerk 127 für die *Loopback-Adresse*. Die Default Route wird bei der Routerkonfiguration verwendet. Über die Loopback-Adresse kann ein Host auf einfache Weise sich selbst adressieren.

- Die *Klasse B* ist für mittelgroße Netzwerke vorgesehen. Die Kombination der beiden höchsten Bits ist bei diesen Adressen immer 10, die verbleibenden 30 Bits legen wieder das Netzwerk (14 Bits) und die Hostadresse (16 Bits) fest:



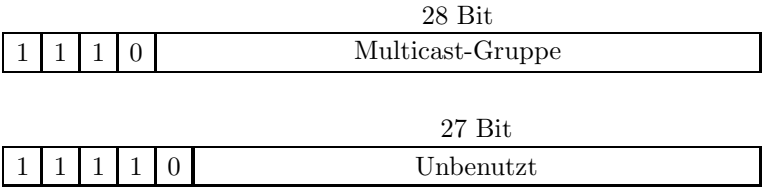
In der üblichen Dezimal-Notation können diesmal die ersten beiden Bytes zusammen als die Netzwerkadresse und die restlichen zwei Bytes als die Hostadresse interpretiert werden. Entsprechend gibt es maximal $2^{14} = 16364$ solcher Klasse-B-Netzwerke, mit jeweils $2^{16} - 2 = 65534$ Hostadressen.

- Die *Klasse C* ist schließlich für kleine Netzwerke vorgesehen. Die Kombination der drei höchsten Bits ist bei diesen Adressen immer 110, und auch hier legen die verbleibenden 29 Bits das Netzwerk (21 Bits) und die Hostadresse (8 Bits) fest:



In der üblichen Dezimal-Notation können diesmal sogar die ersten drei Bytes zusammen als Netzwerkadresse und nur noch das restliche eine Byte als Hostadresse interpretiert werden. Es gibt $2^{21} = 2.097.152$ solcher Klasse-C-Netzwerke, mit jeweils $2^8 - 2 = 254$ Hostadressen.

- Adressen der Klassen D (alle Adressen, die mit der Bit-Kombination 1110 beginnen) und E (alle Adressen, die mit der Bit-Kombination 11110 beginnen) werden nicht als Hostadressen verwendet:



Bei den Adressen der Klasse D handelt es sich um *Multicast-Adressen*, d. h. um Adressen, über die nicht nur ein einzelner Host, sondern gleich eine ganze Gruppe von Hosts erreicht werden kann. Adressen der Klasse E waren von Anfang an für zukünftige Erweiterungen reserviert und werden bis heute nicht verwendet.

In jedem Netzwerk haben zwei Hostadressen eine besondere Bedeutung. Die jeweils erste Hostadresse (Hostadresse 0) bezeichnet das Netzwerk an sich (z. B. 192.168.0.0) und darf nicht einem Host zugewiesen werden. Die letzte Hostadresse (Hostadresse 255) bezeichnet alle Hosts in einem Netzwerk (z. B. 192.168.0.255) und darf ebenfalls nicht einem einzelnen Host zugewiesen werden. Über diese so genannte *Broadcastadresse* kann man ein Datenpaket an alle Hosts in einem lokalen Netzwerk gleichzeitig adressieren.

Als Ergänzung dieses Schemas ist heutzutage eine zusätzliche Einteilung der Hostadresse in eine Subnetzadresse und die eigentliche Hostadresse üblich. Das ist insbesondere für Klasse-B- und Klasse-A-Netzwerke von Bedeutung, da es nicht praktikabel ist, Tausende oder gar Millionen von Hosts an ein einzelnes lokales Rechnernetz anzuschließen.

Innerhalb eines Klasse-B-Netzwerks kann man beispielsweise die für die Hostadresse vorgesehenen 16 Bits hälftig zwischen Subnetzadresse (8 Bits) und Hostadresse (8 Bits) aufteilen und erhält 254 Subnetze mit jeweils 254 Hosts pro Subnetz.

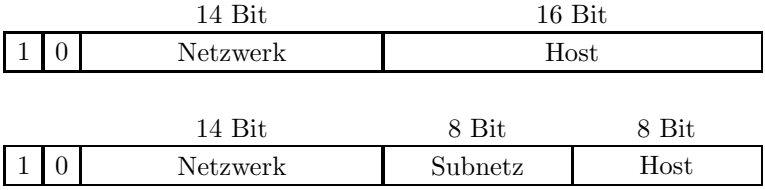


Tabelle 2.1. Adressräume der fünf ursprünglichen Netzwerk-Klassen.

Netzwerk-Klasse	IP-Adressraum
Klasse A	0.0.0.0 bis 127.255.255.255
Klasse B	128.0.0.0 bis 191.255.255.255
Klasse C	192.0.0.0 bis 223.255.255.255
Klasse D	224.0.0.0 bis 239.255.255.255
Klasse E	240.0.0.0 bis 247.255.255.255

Diese Aufteilung erfolgt mit Hilfe der Subnetzmaske, welche neben der eigenen IP-Adresse jedem Host bekannt sein muss. Bei der Subnetzmaske handelt es sich ebenso wie bei einer IP-Adresse um eine 32 Bit bzw. 4 Byte lange Zahl, die als Bitmuster interpretiert wird. Anhand der Subnetzmaske kann der Host für jede IP-Adresse aus dem eigenen Netzwerk feststellen, wo die Subnetzadresse endet und wo die eigentliche Hostadresse beginnt.

Die Subnetzmaske enthält 1-Bits für die Netzwerkadresse und die Subnetzadresse und 0-Bits für die Hostadresse. Um eine Klasse-B-IP-Adresse, wie im obigen Beispiel, hälftig in Subnetzadresse und Hostadresse aufzuteilen, muss demnach die Subnetzmaske 255.255.255.0 (binär: 1111 1111 1111 1111 1111 1111 0000 0000) angegeben werden.

Auch wenn nun bei der Ur-Konfiguration eine Subnetzmaske abgegeben wurde, prüft der sendende Host zuerst, ob die Netzwerkadresse des Empfängers eines IP-Datenpakets mit seiner eigenen übereinstimmt. Ist das nicht der Fall, sendet er das Datenpaket an den nächstgelegenen Router. Stimmen die Netzwerkadressen hingegen überein, berechnet der sendende Host anhand der Subnetzmaske mit Hilfe einer logischen UND-Operation die Subnetzadresse des Empfängers und vergleicht sie mit seiner eigenen. Bei Übereinstimmung kann er das Datenpaket direkt zustellen, bei Nichtübereinstimmung muss er sich der Hilfe eines Routers bedienen.

Für die Konfiguration eines TCP/IP-Hosts wird also mindestens eine IP-Adresse, eine Subnetzmaske und die IP-Adresse des nächstgelegenen Routers benötigt. Für die Konfiguration einer Gruppe von TCP/IP-Hosts wird eine IP-Netzwerkadresse, eine Subnetzmaske und die IP-Adresse des nächstgelegenen Routers benötigt, falls die Gruppe einen Anschluss an andere TCP/IP-Rechnernetze bzw. das Internet benötigt.

Um Überschneidungen zu vermeiden und die Eindeutigkeit von Netzwerkadressen zu wahren[2] wird die Vergabe von Netzwerkadressen international durch die Internet Assigned Numbers Authority (IANA) koordiniert. Die IANA weist dabei Blöcke von IP-Adressen so genannten Regional Internet Registries zu (zuständig für Europa ist Réseaux IP Européens Network Coordination Centre, kurz RIPE NCC). Die wiederum delegieren die eigentliche Vergabe an so genannte Local Internet Registries (in Deutschland z.B Deutsche Telekom AG u. a.).

In der Praxis wird die Netzwerkadresse und die Subnetzmaske häufig vom Zugangsanbieter (Internet Service Provider, kurz ISP) vorgegeben. Dem Administrator steht dann zwar frei, individuell die Zuordnung von Hostadressen zu Hosts festzulegen, die Gesamtanzahl der zu vergebenden Hostadressen steht jedoch von vornherein fest. Im Extremfall bekommt man vom Zugangsanbieter nur eine einzige IP-Adresse, die sich bei jedem Verbindungsaufbau ändert (dynamische IP-Adresse).

[2] Die Eindeutigkeitswahrung im Bezug auf die Hostadressen innerhalb eines Netzwerks liegt hingegen natürlich im Verantwortungsbereich des jeweiligen lokalen Administrators.

2.1.3 Datenpakete und ihr Inhalt

Anwendungsdaten werden in TCP/IP-Rechnernetzen in Form von Datenpaketen verschickt. Die einfachen Pakete der Verbindungsschicht transportieren IP-Pakete der Netzwerkschicht oder deren Fragmente. Die IP-Pakete wiederum transportieren als Nutzlast UDP- oder TCP-Pakete, welche jeweils Teile der eigentlichen Anwendungsdaten tragen.

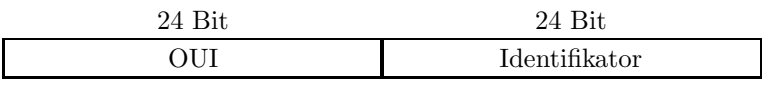
Ethernet-Pakete

Auf welche Weise Datenpakete von Rechner zu Rechner bzw. von Host zu Host innerhalb eines lokalen Rechnernetzes (Local Area Network, kurz LAN) gelangen, hängt von der verwendeten Vernetzungstechnologie ab. Die derzeit wohl am häufigsten anzutreffende Technologie ist Ethernet.

Ethernet-Rechnernetze basieren auf dem Prinzip eines von allen angeschlossenen Hosts geteilten Übertragungsmediums. Jeder Host kann auf dieses geteilte Übertragungsmedium Datenpakete schreiben, welche dann von allen anderen Hosts gelesen werden können.

Jeder Ethernetschnittstelle ist dabei ab Werk eine weltweit eindeutige Ethernetadresse zugeordnet. Ethernetadressen werden oft auch als Hardware-Adressen oder MAC-Adressen bezeichnet (MAC, oder ausgeschrieben Media Access Control, regelt als eine Teilfunktion der Verbindungsschicht den gemeinsamen Zugriff auf das geteilte Übertragungsmedium), wobei Apple die Bezeichnung Ethernet-ID bzw. AirPort-ID gewählt hat.

Jede Ethernetadresse ist 48 Bit bzw. 6 Byte lang und setzt sich aus einem 24 Bit Organizationally Unique Identifier (OUI) und einem eindeutigen, schnittstellenspezifischen 24-Bit-Identifikator zusammen:



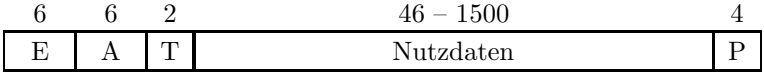
Hersteller von Ethernet-Netzwerkschnittstellen müssen bei dem Institute of Electrical and Electronics Engineers (kurz IEEE) einen OUI beantragen und dafür sorgen, dass alle hergestellten Netzwerkschnittstellen mit unterschiedlichen Identifikatoren versehen werden.

Aufgrund dieses Verfahrens ist es leicht, anhand der Ethernetadresse einer Komponente den Hersteller zu ermitteln. Dazu bietet IEEE die Möglichkeit, über eine Webschnittstelle eine Volltextsuche in der OUI-Registrierungsdatenbank durchzuführen: http://standards.ieee.org/regauth/oui/index.shtml

So lautet beispielsweise die Ethernetadresse (bzw. Ethernet-ID) der integrierten Ethernetschnittstelle eines der PowerBooks des Autors: 00:0a:95:af:30:be. Sucht man nun nach der OUI dieser Adresse in der Registrierungsdatenbank (Eingabe ohne Doppelpunkte als 000a95 erforderlich), erhält man den Eintrag:

```
00-0A-95    (hex)                Apple Computer, Inc.
000A95      (base 16)            Apple Computer, Inc.
                                 1 Infinite Loop
                                 M/S 26-C
                                 Cupertino CA 95014
                                 UNITED STATES
```

Ein Ethernetpaket in dem heute gebräuchlichen Format Ethernet II setzt sich aus der Ethernetadresse des Absenders, der Ethernetadresse des Empfängers, einem Typfeld, den Nutzdaten sowie einem Prüfsummenfeld zusammen:



Das Typfeld besteht aus zwei Bytes und gibt Auskunft über die Art der Nutzdaten. Unterschiedliche Zahlen (Wertebereich dezimal 0–65535 bzw. hexadezimal 0x0000 bis 0xffff) stehen dabei für jeweils unterschiedliche Protokolle. Zugelassen sind u. a. IP bzw. Internet-Protocol (Typ 0x0800), ARP bzw. Address Resolution Protocol (Typ 0x0806), RARP bzw. Reverse Address Resolution Protocol (Typ 0x8035), AppleTalk (Typ 0x809b), AARP bzw. Appletalk Address Resolution Protocol (Typ 0x80f3) und IPv6 (Typ 0x86dd).

In einem Ethernetpaket kann nur eine sehr begrenzte Anzahl von Bytes von Host zu Host transportiert werden. Die Länge der Nutzdaten ist variabel und kann zwischen 46 und 1500 Bytes betragen. Die Anzahl der Bytes, die eine Netzwerktechnologie auf einmal transportieren kann, nennt man in TCP/IP-Netzwerken Maximum Transfer Unit, kurz MTU. Die Größe der MTU beträgt in Ethernet-Netzwerken demnach normalerweise 1500 Bytes.

Jeder an ein Ethernet-Netzwerk angeschlossene Host liest ständig alle auf das Netzwerk geschriebenen Pakete mit. Diejenigen Pakete, die an andere Hosts adressiert sind, werden dabei automatisch verworfen, diejenigen Pakete, deren Empfängeradresse mit der eigenen Ethernetadresse übereinstimmt, werden an übergeordnete Softwareschichten zur weiteren Verarbeitung geleitet.

Obwohl damit ein einfacher Mechanismus zur Übertragung von Daten von Host zu Host bereitsteht, verwendet kaum eine Anwendung direkt Ethernetpakete. Zu groß sind die Einschränkungen. So kann z. B. aufgrund von physikalischen Gesetzmäßigkeiten ein Ethernet-Netzwerk keine beliebigen Ausmaße annehmen. Typische Ethernet-Netzwerke umfassen ein Stockwerk oder ein Gebäude, größere Distanzen müssen hingegen mit Hilfe anderer Vernetzungstechnologien überbrückt werden.

IP-Pakete

Ein IP-Paket ist eine Bytefolge, welche aus einem mindestens 20 Bytes langen Header gefolgt von den eigentlichen Nutzdaten besteht. Der Header beinhaltet unter anderem die IP-Adresse des Absenders und die IP-Adresse des

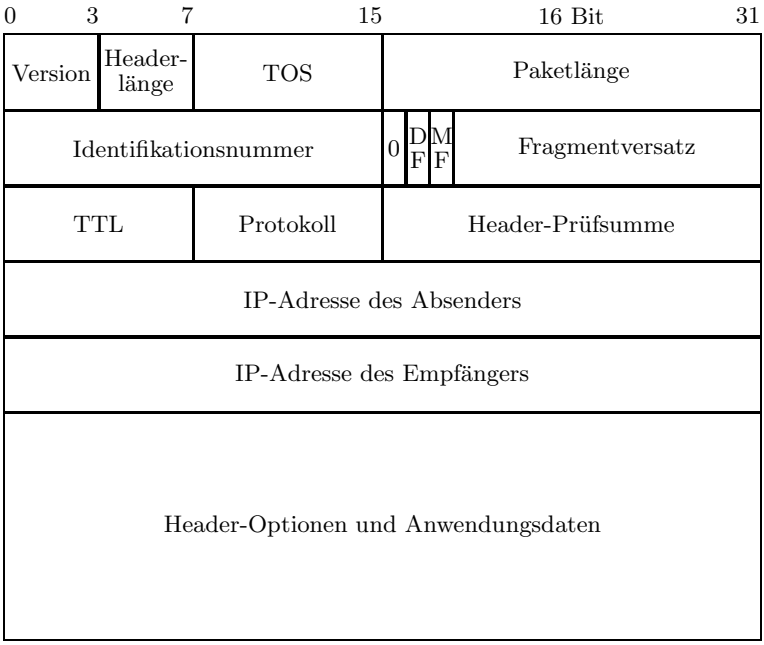


Abb. 2.1. Ein IP-Paket hat einen normalerweise 20 Bytes langen Header. Optionen können ihn auf bis zu 60 Bytes verlängern.

Empfängers. Die maximale Gesamtlänge eines IP-Pakets ist auf 65535 Bytes beschränkt, wobei aber die meisten Netzwerktechnologien ohnehin nicht in der Lage sind Pakete dieser Größe am Stück zu transportieren. Ethernet-Netzwerke haben typischerweise eine MTU von 1500 Bytes, d. h. sie können Pakete von maximal 1500 Bytes Länge transportieren.

Beim Überqueren von mehreren Rechnernetzen in einem Rechnernetzverbund kann es allerdings vorkommen, dass IP-Pakete über Rechnernetze hinweg transportiert werden müssen, die eine MTU haben, welche kleiner ist, als die MTU des Ausgangsrechnernetzes. In diesem Fall muss der Router eine Anpassung vornehmen, indem er jedes IP-Paket in mehrere kleinere Fragmente zerlegt. Jedes Fragment wird wiederum als IP-Paket weiterverschickt und im Header besonders gekennzeichnet. Der Empfänger kann nach Empfang anhand dieser Kennzeichnung zusammengehörige Fragmente wieder zu einem kompletten IP-Paket zusammensetzen. Eine Anwendung, die IP benutzt, merkt von diesen Details allerdings nichts – die Netzwerkschicht kümmert sich selbstständig um diese Dinge.

Abbildung von IP-Adressen auf Ethernetadressen

Um ein IP-Paket mit Hilfe eines Ethernet-Netzwerks an einen anderen Host zu schicken, muss der Absender die Ethernetadresse des Empfängers, bzw.

die Ethernetadresse des Routers kennen. Um die IP-Adresse des Empfängers in eine Ethernetadresse zu konvertieren bedient sich der sendende Host des *Addresss Resolution Protocols*, kurz ARP.

Dazu sendet er eine aus einem einzelnen Ethernet-Paket bestehende ARP-Anfrage an alle an das lokale Netzwerk angeschlossenen Hosts. Ein solcher Ethernet-*Broadcast*, bei dem alle angeschlossenen Hosts das Paket empfangen können, ist möglich, indem als Ethernetadresse des Empfängers die spezielle Broadcast-Adresse ff:ff:ff:ff:ff:ff angegeben wird.

Jeder TCP/IP-Host verfügt über Software, die permanent aktiv ist, und solche Anfragen mitliest. Falls es sich bei der angefragten IP-Adresse um die eigene handelt, antwortet der Host dem Absender der ARP-Anfrage mit einem Ethernet-Paket, mit dem es die eigene Ethernetadresse mitteilt.

Damit nicht vor der Übermittlung eines jeden IP-Pakets eine ARP-Anfrage verschickt werden muss, werden die so ermittelten Ethernetadressen im ARP-*Cache* gespeichert. Die in diesem Cache zwischengespeicherten Werte werden normalerweise für 20 Minuten als gültig betrachtet, bevor sie durch eine erneute ARP-Anfrage aktualisiert werden.

Der Inhalt des ARP-Caches kann mit dem Befehl arp ausgelesen werden. Die Option -a veranlasst die Ausgabe aller Einträge aus diesem Zwischenspeicher:

```
$ arp -a
? (192.168.0.1) at 0:50:7f:0:10:45 on en0 [ethernet]
? (192.168.0.2) at 0:3:93:19:69:6a on en0 [ethernet]
```

Mit der Option -d kann man gezielt Einträge aus dem Cache entfernen (root-Rechte erforderlich). Die Kombination der Option -d mit der Option -a entfernt alle Einträge aus dem ARP-Cache:

```
$ arp -a
? (192.168.0.1) at 0:50:7f:0:10:45 on en0 [ethernet]
? (192.168.0.2) at 0:3:93:19:69:6a on en0 [ethernet]
$ sudo arp -d -a
192.168.0.1 (192.168.0.1) deleted
192.168.0.2 (192.168.0.2) deleted
$ arp -a
$
```

Der ARP-Cache wird automatisch wieder gefüllt, sobald TCP/IP-Anwendungen gestartet werden, die IP-Pakete verschicken. Eine der einfachsten TCP/IP-Anwendungen ist der Diagnosebefehl ping. Dieser Befehl schickt einfache IP-Pakete, so genannte *ICMP*[3] *Echo Requests*, an den Empfänger. Der Empfänger antwortet darauf mit *ICMP Echo Reply* – so dass auf einfache Weise festgestellt werden kann, ob ein bestimmter Host erreichbar ist.

[3] Das ICMP-Protokoll (Internet Control Message Protocol) definiert neben den Nachrichtentypen Echo Request und Echo Reply eine Reihe weiterer Nachrichtentypen, die in IP-Pakete eingebettet werden können. Die meisten davon dienen der Fehlerübermittlung (z. B. Host nicht erreichbar u.ä.).

Mit der Option -c kann die Anzahl der von ping verschickten Pakete limitiert werden (normalerweise sendet ping ein Paket pro Sekunde, bis der Befehl z. B. durch CTRL-C abgebrochen wird). Wir können beobachten, dass bereits ein einzelnes Paket einen Eintrag im ARP-Cache erzeugt:

```
$ arp -a
$ ping -c 1 192.168.0.2
PING 192.168.0.2 (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: icmp_seq=0 ttl=64 time=0.552 ms

--- 192.168.0.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.552/0.552/0.552 ms
$ arp -a
? (192.168.0.2) at 0:3:93:19:69:6a on en0 [ethernet]
```

Hätten wir nun statt eines direkt erreichbaren Hosts (im obigen Beispiel 192.168.0.2) einen Host verwendet, der nur mittelbar über den Router erreichbar ist, würde die Ausgabe allerdings etwas anders aussehen:

```
$ arp -a
$ ping -c 1 17.112.152.32
PING 17.112.152.32 (17.112.152.32): 56 data bytes
64 bytes from 17.112.152.32: icmp_seq=0 ttl=54 time=226.367 ms

--- 17.112.152.32 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 226.367/226.367/226.367 ms
$ arp -a
? (192.168.0.1) at 0:50:7f:0:10:45 on en0 [ethernet]
```

Diesmal hat der sendende Host offenbar keine ARP-Anfrage für die IP-Adresse 17.112.152.32 des Zielrechners, sondern für eine völlig andere Adresse (192.168.0.1) gesendet. Der Grund: anhand der IP-Adresse des Zielrechners konnte der sendende Host sofort erkennen, dass der Zielrechner nur über den Router erreichbar ist, und hat deshalb die Ethernetadresse des Routers angefragt[4].

User Datagram Protocol

Anwendungen erzeugen in der Regel IP-Pakete nicht direkt, sondern bedienen sich der Protokolle UDP oder TCP, um ihre Daten auszutauschen. Die Einbettung von UDP- und TCP-Paketen in IP-Pakete erfolgt automatisch mit Hilfe der TCP/IP-Software. Dabei wird jedes IP-Paket im Header mit einer Protokollnummer als UDP- oder TCP-Paket gekennzeichnet, damit der Empfänger beide Paketarten leicht unterscheiden kann.

Neben UDP und TCP kann eine Vielzahl von anderen Protokollen in IP-Paketen transportiert werden. In Mac OS X 10.4 werden die gültigen

[4] Anhand der Ethernetadresse bzw. der OUI (00:50:7f, siehe Abschn. 2.1.3) kann man sehen, dass es sich bei dem Router im Beispiel um ein Produkt der Firma Draytek handelt.

Protokollnummern, ähnlich wie in anderen Unix-basierten Systemen, in der Datei /etc/protocols gelistet (Ausgabe gekürzt):

```
$ cat /etc/protocols
#
# Internet-Protocols
#
# $FreeBSD: src/etc/protocols,v 1.14 2000/09/24 11:20:27 asmodai Exp $
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# See also http://www.isi.edu/in-notes/iana/assignments/protocol-numbers
#
ip      0       IP              # internet-Protocol, pseudo protocol
                                  number
#hopopt 0       HOPOPT          # hop-by-hop options for ipv6
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group management protocol
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP
                                  (officially "IP")
st2     5       ST2             # ST2 datagram mode (RFC 1819)
tcp     6       TCP             # transmission control protocol
cbt     7       CBT             # CBT, Tony Ballardie
                                  <A.Ballardie@cs.ucl.ac.uk>
egp     8       EGP             # exterior gateway protocol
igp     9       IGP             # any private interior gateway
                                  (Cisco: for IGRP)
bbn-rcc 10      BBN-RCC-MON     # BBN RCC Monitoring
nvp     11      NVP-II          # Network Voice Protocol
pup     12      PUP             # PARC universal packet protocol
argus   13      ARGUS           # ARGUS
emcon   14      EMCON           # EMCON
xnet    15      XNET            # Cross Net Debugger
chaos   16      CHAOS           # Chaos
udp     17      UDP             # user datagram protocol
mux     18      MUX             # Multiplexing protocol
dcn     19      DCN-MEAS        # DCN Measurement Subsystems
hmp     20      HMP             # host monitoring protocol
prm     21      PRM             # packet radio measurement protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
trunk-1 23      TRUNK-1         # Trunk-1
trunk-2 24      TRUNK-2         # Trunk-2
leaf-1  25      LEAF-1          # Leaf-1
leaf-2  26      LEAF-2          # Leaf-2
rdp     27      RDP             # "reliable datagram" protocol
irtp    28      IRTP            # Internet Reliable Transaction Protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol Class 4
netblt  30      NETBLT          # Bulk Data Transfer Protocol
mfe-nsp 31      MFE-NSP         # MFE Network Services Protocol
merit-inp       32      MERIT-INP       # MERIT Internodal Protocol
sep     33      SEP             # Sequential Exchange Protocol
3pc     34      3PC             # Third Party Connect Protocol
idpr    35      IDPR            # Inter-Domain Policy Routing Protocol
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
...
fc      133     FC              # Fibre Channel
#       134-254                 # Unassigned
divert  254     DIVERT          # Divert pseudo-protocol [non IANA]
#       255                     # Reserved
```

Jede Zeile von /etc/protocols beinhaltet drei Einträge: den offiziellen Protokollnamen, die Protokollnummer und ein Protokollalias. Mit # werden Kommentare eingeleitet.

Aus dieser Datei kann man entnehmen, dass 1 die Protokollnummer von ICMP, 6 die Protokollnummer von TCP und 17 die Protokollnummer von UDP ist. Neben Protokollen aus der TCP/IP-Protokollfamilie lassen sich natürlich auch beliebige andere Daten mit Hilfe von IP transportieren. Insbesondere lassen sich auch Pakete aus anderen Protokollfamilien problemlos in IP kapseln: so ist z. B. die Protokollnummer 37 dem Datagram Delivery Protocol (kurz DDP) zugeordnet, das zur AppleTalk-Protokollfamilie (siehe Abschn. 2.2) gehört.

UDP ist im Vergleich zu TCP das wesentlich einfachere Protokoll. Die Anwendung muss selbst die zu sendenden Daten in Pakete aufteilen und sich um die Behandlung von verloren gegangenen Paketen kümmern. Jedes UDP-Paket wird unmittelbar in ein IP-Paket eingebettet. Der zusätzliche UDP-Header beinhaltet dabei lediglich die Port-Nummer des Absenders, die Port-Nummer des Empfängers, die Länge des Pakets sowie eine Prüfsumme (Abb. 2.2).

Mit Hilfe der vom Absender des Pakets berechneten Prüfsumme kann der Empfänger beschädigte UDP-Pakete erkennen und verwerfen. Die TCP/IP-Software generiert in diesem Fall jedoch keine Fehlermeldung. Ein beschädigtes UDP-Paket wird demnach so behandelt, als wäre es nie zugestellt worden. Es liegt im Verantwortungsbereich der Anwendungssoftware, die verloren gegangene Nutzlast bei Bedarf erneut beim Absender anzufordern.

Bei den Port-Nummern handelt es sich um 16-Bit-Zahlen (0 bis 65535), welche den jeweils sendenden und den empfangenen Prozess identifizieren. Port-Nummern werden sowohl von UDP als auch von TCP verwendet und

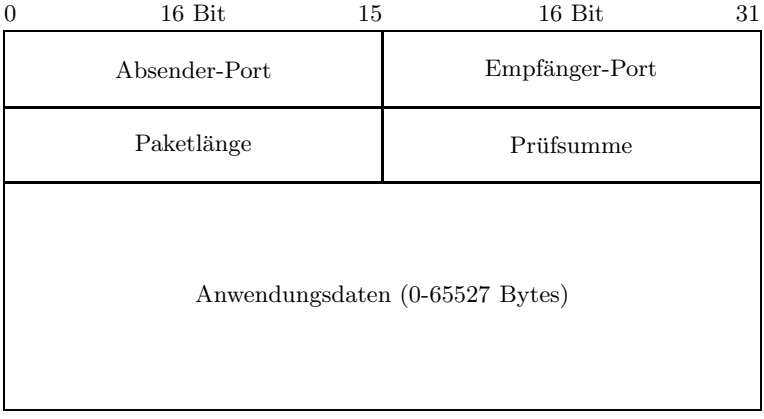


Abb. 2.2. Ein UDP-Paket besteht aus einem Acht-Byte-Header und einem Anwendungsdatenfeld variabler Länge.

tragen der Tatsache Rechnung, dass auf einem Host normalerweise mehrere Prozesse ablaufen können, die über TCP/IP kommunizieren möchten. Ohne Port-Nummern wäre eine Zuordnung von Paketen zu Prozessen nicht möglich.

Eine Kommunikation über UDP oder TCP mit einem Prozess auf einem anderen Host ist also nur dann möglich, wenn sowohl die IP-Adresse des Hosts als auch die vom entfernten Prozess benutzte Port-Nummer bekannt ist.

Um den Konfigurations- und Kommunikationsaufwand für die Anwender zu minimieren, sind viele Port-Nummern bestimmten Anwendungen fest zugeordnet. Beispielsweise verwendet das Apple Filesharing Protocol AFP (z. B. Personal File Sharing in Mac OS X) die Port-Nummer 548 und das Hyper-Text Transfer Protocol HTTP (z. B. Personal Web Sharing in Mac OS X) die Port-Nummer 80.

Auf diese Weise muss ein Anwender, der auf einen beliebigen (IP-fähigen) AFP-Server zugreifen möchte, nur noch dessen IP-Adresse kennen und muss sich keine Gedanken über die korrekte Port-Nummer machen. Ebenso ist es bei der Eingabe von URLs in einem Web-Browser normalerweise nicht nötig, eine Port-Nummer anzufügen, da erwartet werden kann, dass ein Web-Server normalerweise die Standard-Port-Nummer 80 verwendet.

Man unterscheidet zwischen Wohl-Bekannten (Well Known), Registrierten (Registered) und Dynamischen Ports. Wohl-Bekannte Ports sind Ports mit Nummern im Bereich von 0 bis 1023. Damit ein Server-Prozess eine solche Port-Nummer reservieren kann, muss er mit root-Rechten gestartet worden sein. Registrierte Ports sind Ports mit Nummern im Bereich von 1.024 bis 49.151. Diese Port-Nummern sind zwar ebenso wie Wohl-Bekannte Ports bestimmten Anwendungen zugeordnet, root-Rechte sind für die Verwendung dieser Ports jedoch nicht erforderlich. Dynamische Ports sind schließlich alle verbleibenden Ports mit Nummern im Bereich 49.152 bis 65.535.

Die Zuordnung von Port-Nummern zu Anwendungen wird von IANA koordiniert und kann unter Mac OS X 10.4 der Datei /etc/services entnommen werden (auch hier aus Platzgründen die Ausgabe nur gekürzt):

```
$ cat /etc/services
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single
  well-known
# port number for both TCP and UDP; hence, most entries here have
  two entries
# even if the protocol doesn't support UDP operations.
#
# The latest IANA port assignments can be gotten from
#
#       http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through
```

```
  65535
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eric Exp $
#       From: @(#)services      5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#
rtmp              1/ddp    #Routing Table Maintenance Protocol
tcpmux            1/udp     # TCP Port Service Multiplexer
tcpmux            1/tcp     # TCP Port Service Multiplexer
#                           Mark Lottor <MKL@nisc.sri.com>
nbp               2/ddp    #Name Binding Protocol
compressnet       2/udp     # Management Utility
compressnet       2/tcp     # Management Utility
compressnet       3/udp     # Compression Process
compressnet       3/tcp     # Compression Process
#                           Bernie Volz <VOLZ@PROCESS.COM>
echo              4/ddp    #AppleTalk Echo Protocol
#                 4/tcp    Unassigned
#                 4/udp    Unassigned
rje               5/udp     # Remote Job Entry
rje               5/tcp     # Remote Job Entry
#                           Jon Postel <postel@isi.edu>
zip               6/ddp    #Zone Information Protocol
#                 6/tcp    Unassigned
#                 6/udp    Unassigned
echo              7/udp     # Echo
echo              7/tcp     # Echo
#                           Jon Postel <postel@isi.edu>
...
#                            Nicholas J Howes <nick@ghostwood.org>
#               48557-49150 Unassigned
#               49151       IANA Reserved
```

Jede Zeile von /etc/services enthält die offizielle Anwendungsbezeichnung, die Port-Nummer, die Bezeichnung des verwendeten Protokolls (meistens UDP oder TCP, unter Mac OS X manchmal auch DDP) sowie optional ein Anwendungsalias. Ebenso wie in /etc/protocols werden Kommentare mit einem # eingeleitet.

Die Angabe des verwendeten Protokolls ist erforderlich, weil Port-Nummern nur innerhalb eines Protokolls eindeutig sind. Ein und dieselbe Port-Nummer kann je nach Protokoll unterschiedlichen Anwendungen zugeordnet sein. Aus der Kombination der Protokoll- und der Port-Nummer kann die Anwendung jedoch immer eindeutig ermittelt werden.

Transmission Control Protocol

Gegenüber UDP ist TCP etwas komplexer, sorgt dafür jedoch selbstständig für die Aufteilung der zu übertragenden Daten in Pakete (TCP-*Segmente*), deren Zustellung in der korrekten Reihenfolge sowie für die erneute Übertragung von unterwegs verloren gegangenen oder beschädigten Paketen. Dabei arbeitet TCP verbindungsorientiert und bidirektional. Das bedeutet, dass man nicht einfach Pakete drauflosverschicken kann, sondern zuerst eine TCP-

Verbindung zwischen den beiden Kommunikationspartnern (i. Allg. Prozessen) aufbauen muss, über die dann in Folge Datenströme in beide Richtungen verschickt werden können.

Im Vergleich zu UDP weist TCP einen etwas längeren, 20 Bytes umfassenden eigenen Header auf. Dieser Header beinhaltet neben der Port-Nummer des Absenders und der Port-Nummer des Empfängers einige weitere Informationen, darunter unter anderem eine 32-Bit-*Sequenznummer*, eine 32-Bit-*Bestätigungsnummer*, die 16-Bit-Fenstergröße, eine Prüfsumme für den Header und die Nutzlast sowie eine Reihe von Flags (Abb. 2.3).

TCP verwendet die Sequenznummern, um die korrekte Reihenfolge von empfangenen Segmenten zu garantieren, die Bestätigungsnummern, um dem Absender das letzte korrekt empfangene Segment zu bestätigen und die Fenstergröße, um die Größe des Empfangspuffers anzuzeigen. Mit Hilfe der Checksumme kann der Empfänger feststellen, ob ein TCP-Segment beim Transport beschädigt worden ist.

Zu Beginn einer Datenübertragung wird eine logische Host-zu-Host-Verbindung aufgebaut, über die Datenströme in beide Richtungen übertragen werden können. Dazu benutzt TCP ein Drei-Wege-Handshake – es werden drei Segmente ausgetauscht, bevor die eigentliche Datenübertragung beginnen kann.

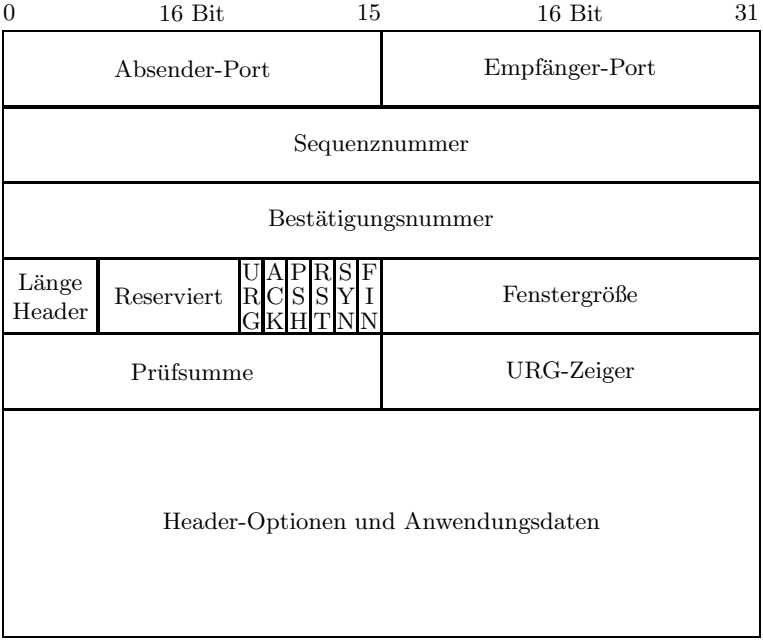


Abb. 2.3. Ein TCP-Paket hat einen normalerweise 20 Bytes langen Header. Optionen können ihn auf bis zu 60 Bytes verlängern.

Der Initiator der Verbindung sendet dazu ein erstes TCP-Segment mit gesetztem SYN-Flag (Synchronisation der Sequenznummern). Mit diesem Segment zeigt der Initiator den Wunsch, eine neue TCP-Verbindung aufzubauen, an und teilt dem Empfänger die Start-Sequenznummer und die Größe seines Empfangspuffers mit. Der Empfänger schickt daraufhin dem Initiator ein Segment mit gesetztem SYN-Flag und gesetztem ACK-Flag (Acknowledge/Bestätigung) zurück, indem er mit Hilfe der Bestätigungsnummer die Sequenznummer des Initiators bestätigt und gleichzeitig die eigene Start-Sequenznummer (TCP-Verbindungen sind ja bidirektional) und die eigene Empfangspuffergröße anzeigt. Der Initiator sendet im dritten und letzten Schritt des Verbindungsaufbaus dann wiederum ein Segment zurück, welches den Empfang des Segments bestätigt (gesetztes ACK-Flag, aber ohne das SYN-Flag) und bereits auch Nutzdaten enthalten kann.

Über eine bestehende TCP-Verbindung kann in jede Richtung ein aus einer Folge von Bytes bestehender Datenstrom übertragen werden. Die Bytes dieser beiden Datenströme werden dabei automatisch fortlaufend durchnummeriert und in Segmente aufgeteilt. Diese Nummerierung beginnt mit den mit Hilfe der beiden Synchronisationssegmente während des Verbindungsaufbaus ausgetauschten Start-Sequenznummern. Sie beginnt üblicherweise nicht bei 0, sondern aus Sicherheitsgründen mit einer Zufallszahl.

Im Header eines jeden TCP-Segments steht die Sequenznummer des ersten dort transportierten Bytes, so dass auf Empfängerseite die segmentweise übertragenen Bytes wieder zu einem Datenstrom unter Einhaltung der korrekten Reihenfolge zusammengesetzt werden können.

Der Empfänger eines TCP-Segments kann den korrekten Empfang des Segments bestätigen, indem er an den Absender ein TCP-Segment mit gesetztem ACK-Flag und einer Bestätigungsnummer schickt, die der Sequenznummer des nächsten erwarteten Bytes entspricht. Auf diese Weise kann der Empfänger mehrere TCP-Segmente auf einmal bestätigen.

Angenommen, Host A schickt zwei TCP-Segmente mit der Länge von jeweils 1460 Bytes[5] an Host B und als Start-Sequenznummer ist 0 vereinbart worden. In diesem Fall wäre die Sequenznummer im ersten Segment $0+1 = 1$ und die Sequenznummer im zweiten Segment $1 + 1460 = 1461$. Host B muss nach dem Empfang beider Segmente nicht etwa zuerst das erste Segment mit der Bestätigungsnummer 1461 quittieren, sondern kann beide Segmente mit einer einzigen Bestätigung – mit Bestätigungsnummer $1461 + 1460 = 2921$ – quittieren.

Der Absender muss allerdings nicht auf die Bestätigung warten. Anhand der in jedem TCP-Segment übermittelten Fenstergröße kennt der Absender zu jedem Zeitpunkt die Größe des Empfangspuffers des Empfängers. Er kann

[5] 1460 Bytes ist die maximale Segmentgröße bei einer MTU von 1500 Bytes (also z. B. bei Verwendung von Ethernet als Vernetzungstechnologie), da von den 1500 Bytes jeweils 20 Bytes für den IP-Header und den TCP-Header abgezogen werden müssen.

so lange TCP-Segmente verschicken, wie die Anzahl der noch nicht bestätigten Bytes kleiner ist als die Empfangspuffergröße. Bei einer maximalen Fenstergröße von $2^{16} - 1 = 65535$ Bytes und einer Segmentgröße von 1460 Bytes können bis zu 45 Segmente verschickt werden, bis es einer Bestätigung des Empfängers bedarf.

Treffen keine Bestätigungen ein, erfolgt eine erneute Übermittlung der Segmente. Auf diese Weise stellt TCP sicher, dass keine Daten verloren gehen können, solange zumindest ein Teil der Segmente ihr Ziel erreichen kann.

2.1.4 Private Netze

Sehr oft stehen offiziell von der IANA zugeordnete und damit öffentliche IP-Adressen gar nicht in ausreichender Anzahl zur Verfügung. Gerade im privaten Bereich, aber auch bei vielen kleineren Unternehmen, wird die Verbindung zu anderen Netzwerken über einen Zugangsanbieter realisiert, der lediglich eine einzige, dazu noch veränderliche[6] IP-Adresse bereitstellt.

Manchmal möchte man auch gar keine Verbindungen zu Hosts außerhalb des eigenen Netzwerks ausbauen – sei es, weil man ein Netzwerk zu reinen Ausbildungs- und Testzwecken aufbaut, oder weil man mit einem völlig abgeschotteten Netzwerk aus Sicherheitsgründen arbeiten möchte oder muss.

In beiden Fällen kann man sich aus einem Pool von speziell für einen solchen Einsatz von der IANA reservierten *privaten* Netzwerkadressen bedienen:

- einem A-Klasse-Netzwerk: 10.0.0.0
- einem von 16 B-Klasse-Netzwerken: 172.16.0.0. bis 172.172.31.0.0
- einem von 256 C-Klasse-Netzwerken: 192.168.0.0 bis 192.168.255.0

Sehr gängig ist das C-Klasse-Netzwerk 192.168.0.0. In diesem Netzwerk gibt es 254 Host-Adressen (192.168.0.1 bis 192.168.0.254) – eine Zahl, welche für die meisten Privatanwender und kleine Unternehmen ausreicht.

Neben 192.168.0.0 kann auch jede andere der oben genannten Netzwerkadressen ohne Genehmigung von der IANA einfach verwendet werden. Besondere Vorkehrungen müssen erst dann getroffen werden, wenn von solchen Adressen aus nun doch auf Hosts außerhalb des privaten Netzwerks zugegriffen werden soll. Da jeder Administrator diese Adressen ohne Rücksprache mit anderen einfach verwenden darf, ist im Gegensatz zu öffentlichen IP-Adressen keine Eindeutigkeit gegeben. Die Router der Zugangsanbieter und andere Router im öffentlichen Bereich des Internets unternehmen daher auch gar keinen Versuch, mit solchen Adressen versehene Pakete überhaupt zuzustellen, sondern verwerfen diese sofort.

[6] Selbst bei Pauschaltarifen, neudeutsch *Flatrates*, wird meist nach 24 Stunden die Verbindung kurz getrennt und nach Wiedereinwahl eine neue IP-Adresse vergeben.

Im Wesentlichen existieren zwei Möglichkeiten um dieses Problem zu umgehen: die Nutzung von Stellvertreteranwendungen (engl. Proxy) und die automatische Adressübersetzung (engl. Network Address Translation, kurz NAT).

Beide Verfahren erfordern einen Host, der als Schnittstelle zwischen dem öffentlichen Bereich und dem privaten Bereich agiert. Dieser Host muss ähnlich wie ein Router über zwei Netzwerkschnittstellen verfügen: eine private Schnittstelle, welche mit einer der privaten IP-Adressen versehen wird, sowie eine öffentliche Schnittstelle, der die vom Zugangsanbieter erhaltene, öffentliche IP-Adresse zugewiesen wird. Oft wählt man für die private Schnittstelle dieses Hosts aus Gründen der Übersicht die niedrigste (z. B. 192.168.0.1) oder die höchste (z. B. 192.168.0.254) IP-Adresse des verwendeten Netzwerks. Für die eigentliche Funktion spielt die Auswahl der IP-Adresse natürlich keine Rolle und dient nur als Gedächtnisstütze.

Anders als ein „normaler“ Router kann dieser Host die empfangenen Pakete jedoch nicht einfach weiterleiten. Die aus dem privaten Bereich kommenden Pakete könnten zwar an Hosts im öffentlichen Bereich zugestellt werden, auf eine Antwort würden die Hosts im privaten Bereich mangels einer öffentlichen Absenderadresse aber vergeblich warten. Andersherum sind an den privaten Bereich adressierte Pakete aus dem öffentlichen Bereich nicht zu erwarten, da öffentliche Router, wie bereits erwähnt, solche Pakete mangels Eindeutigkeit der Zieladresse normalerweise sofort verwerfen.

Einen Ausweg bietet die Installation von Proxy-Anwendungen, also von Stellvertreteranwendung auf diesem Host. Proxy-Anwendungen können Anfragen von Hosts aus dem privaten Bereich an Hosts im öffentlichen Bereich weiterleiten und geben sich selbst als Absender aus. Antwortet ein öffentlicher Host auf eine solche Anfrage, muss die Proxy-Anwendung diese Antwort wiederum an den privaten Host weiterleiten können.

Proxy-Anwendungen sind anwendungsspezifisch, d. h. sie unterstützen in der Regel nur eine einzige Anwendung. Ein typisches Beispiel für eine Proxy-Anwendung sind Web-Proxies: Diese Programme leiten Webseiten-Abrufe von privaten Hosts an die jeweiligen Web-Server weiter, nehmen die Webseiten von diesen entsprechend entgegen und leiten sie an die Web-Browser der Anwender weiter. Als nützliche Zusatzfunktion speichern sie bereits abgerufene Web-Seiten in einem lokalen Zwischenspeicher (engl. *Cache*) und verkürzen damit die Ladezeiten.

Proxy-Anwendungen sind nicht völlig transparent, also nicht unsichtbar. Die Entwickler müssen den möglichen Proxy-Betrieb bereits beim Entwurf berücksichtigen und entsprechende Funktionen in den Client-Anwendungen implementieren. Der Anwender muss die IP-Adresse des Proxy-Hosts kennen und seine Anwendung entsprechend konfigurieren (Abb. 2.4).

Während Proxy-Anwendungen zwischen dem öffentlichen und dem privaten Bereich – im Sinne des in Abschn. 2.1.1 vorgestellten Schichtenmodells – innerhalb der Anwendungsschicht vermitteln, arbeitet die automati-

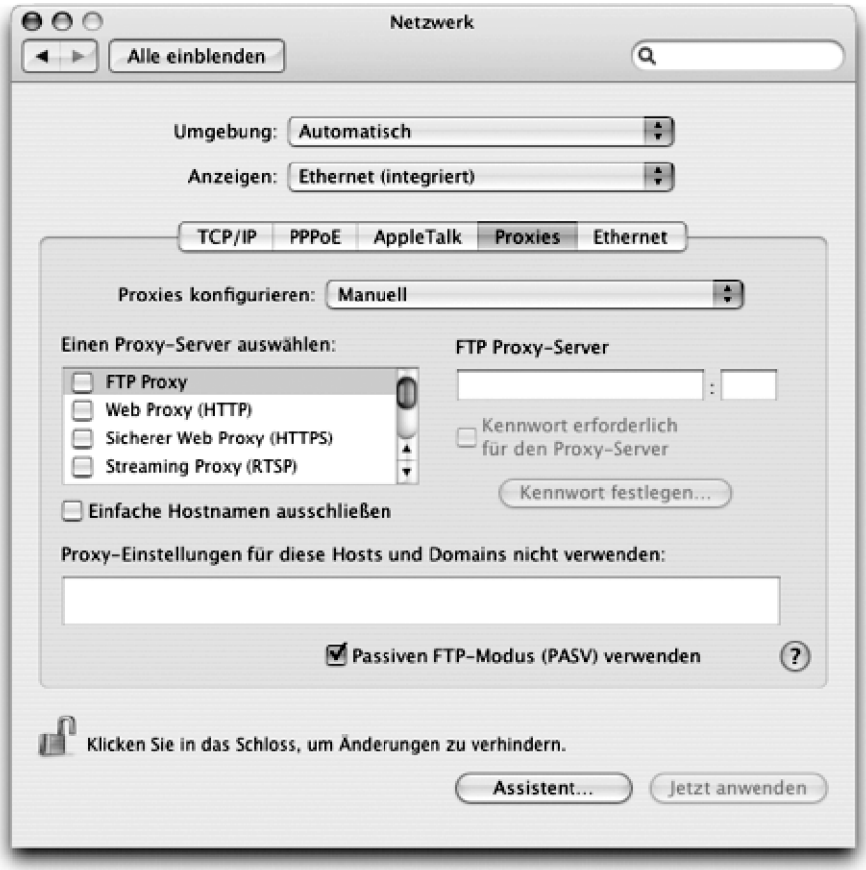


Abb. 2.4. In Mac OS X kann der Anwender die Proxy-Einstellungen zwar an einer zentralen Stelle vornehmen, die Beachtung dieser Einstellung liegt aber naturgemäß in der Verantwortung der Anwendungsprogrammierer. Anders als NAT sind Proxy-Server für die Anwendungen nicht transparent und müssen bei der Anwendungsentwicklung berücksichtigt werden.

sche Adressenübersetzung innerhalb der Transportschicht und ist daher anwendungsunabhängig und nahezu völlig transparent bzw. für den Anwender unsichtbar. Dabei wird der Host, der die automatische Adressübersetzung durchführt, von den privaten Hosts als Router verwendet (man spricht dabei von einem NAT-Router).

Das Grundprinzip ist dabei sehr einfach und am besten an einem konkreten Beispiel nachzuvollziehen. Nehmen wir an, dass der Anwender auf einem privaten Host mit der IP-Adresse 192.168.0.2 den Web-Browser Safari startet und die Apple Homepage betrachten möchte. Dazu gibt er in der Adresszeile die DNS-Adresse *www.apple.de* ein, die von ihm unbe-

merkt sofort in die IP-Adresse 17.254.3.153 übersetzt wird (siehe auch Abschn. 2.1.5).

Safari würde nun versuchen, eine TCP-Verbindung zum öffentlichen Host 17.254.3.153 aufzubauen. Dabei würde Safari als Ausgangspunkt der Verbindung einen dynamischen Port wählen, der nicht Verwendung ist, also z. B. 49.152. Als Endpunkt der Verbindung auf dem öffentlichen Host dient in diesem Fall Port 80, der als Wohl-Bekannter Port für den Dienst HTTP (der zum Abruf von Web-Seiten benutzt wird) in der Datei /etc/services definiert ist.

Da sich der Empfänger 17.254.3.153 außerhalb des lokalen Netzwerks (192.168.0.0) befindet, werden die TCP-Segmente an den designierten Router zur Weiterleitung geschickt. Diese Pakete enhalten u. a. folgende Daten: Absender-IP-Adresse 192.168.0.2, Absender-Port 49.152, Empfänger-IP-Adresse 17.254.3.153 und Empfänger-Port 80.

Wie bereits diskutiert, kann der Router die Pakete in diesem speziellen Fall nicht einfach an den nächsten Router weiterleiten. Da der Absender über keine öffentliche IP-Adresse verfügt, würden die Bestätigungspakete des Empfängers ihn niemals erreichen können. Eine TCP-Verbindung kann auf diese Weise also nie zustande kommen.

Bei aktiver Adressübersetzung modifiziert der NAT-Router deshalb sowohl die ausgehenden als auch die eingehenden Pakete nach einem festgelegten Verfahren.

Bei ausgehenden Paketen ersetzt er die IP-Adresse des Absenders durch die eigene *öffentliche* Adresse, z. B. 84.151.70.223. Gleichzeitig ersetzt er die vom Absender gewählte Port-Nummer durch eine freie Port-Nummer aus einem eigenen Pool von Ports, z. B. durch 35.000. Das so veränderte Paket leitet er an den Empfänger.

Die vorgenommenen Ersetzungen werden in einer Tabelle festgehalten. Diese Tabelle ermöglicht die Zuordnung von Antworten des öffentlichen Hosts zu den privaten Hosts. Ein Tabelleneintrag für unser Beispiel müsste also die Absender-IP-Adresse 192.168.0.2, die Absender-Port-Nummer 49.152, die eingesetzte Port-Nummer 35.000 und die Empfänger-IP-Adresse enthalten.

Mit Hilfe dieser Tabelle kann der NAT-Router von außen eingehende Pakete auf sehr einfache Weise den privaten Hosts zuordnen. Der Router muss dazu lediglich die Ziel-Port-Nummern inspizieren und mit den Einträgen in seiner Tabelle vergleichen. Fehlt ein passender Eintrag, wird das eingehende Paket, sofern es nicht an einen lokalen, also auf dem Router selbst ablaufenden Prozess zugestellt werden kann, verworfen. Passt ein Tabelleneintrag, dann kann diesem Eintrag sowohl die private IP-Adresse als auch die dazugehörige Portnummer entnommen werden. Der Router kann die Empfängeradresse und die Portnummer deshalb einfach ersetzen und an den internen Empfänger weiterleiten.

Mit diesem Verfahren können private Hosts zeitgleich auf die unterschiedlichsten Dienste im öffentlichen Bereich zugreifen. Dabei entstehen selbst dann keine Konflikte, wenn zwei private Hosts, die gleichen Absender-Ports

verwendend, auf ein und denselben öffentlichen Host und dort auf ein und dieselbe Empfänger-Port-Nummer zugreifen möchten. Da der Router die Port-Nummern der Absender durch eindeutige Ersatznummern aus seinem eigenen Pool ersetzt, ist auch in diesem Fall die eindeutige Zuordnung der Antworten möglich.

Dieser Ablauf ist für die meisten Anwendungen völlig transparent und bedarf keinerlei Konfiguration auf Seiten des privaten Hosts oder des NAT-Routers. Nur für den umgekehrten Fall, wenn also öffentlichen Hosts der Zugriff auf interne Dienste ermöglicht werden soll, muss die Tabelle des NAT-Routers vom Administrator um einen entsprechenden festen Eintrag ergänzt werden.

In einem solchen festen Eintrag wird ein bestimmter Port des NAT-Routers permanent an einen bestimmten privaten Host umgeleitet. Auf diese Weise kann ein Web-Server, der auf einem privaten Host läuft, der Öffentlichkeit zugänglich gemacht werden. Dazu muss man lediglich einen Eintrag anlegen, der dem NAT-Router mitteilt, dass Port-Nummer 80 an den entsprechenden privaten Host, z. B. wieder 192.168.0.2, weitergeleitet werden soll.

2.1.5 DNS

Die bisher behandelte Identifikation von Hosts auf der Basis ihrer IP-Adressen ist zwar aus technischer Sicht völlig ausreichend, für Menschen jedoch im Alltagsgebrauch nicht allzu komfortabel. Da es den meisten von uns leichter fällt sich Namen statt Zahlenkombinationen zu merken, gehören zur TCP/IP-Protokollfamilie auch Verfahren, um IP-Adressen Namen zuzuordnen.

Das ältere, tabellengestützte Verfahren bedient sich dabei einer einfachen Textdatei, welche Hostnamen und die dazugehörigen IP-Adressen enthält. Das neuere, datenbankgestützte Verfahren bedient sich einer verteilten Datenbank.

Im tabellengestützten Verfahren erfolgen die Zuordnungen anhand von Einträgen in einer Textdatei – auf Unix-basierten Systemen normalerweise /etc/hosts. Auch auf einem Mac OS X 10.4-System findet man diese Datei, obgleich sie in der Standardkonfiguration lediglich in der frühen Startphase verwendet wird:

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
```

Anhand dieser rudimentären Hosts-Datei kann man das Format der Einträge leicht erkennen. Ein Eintrag entspricht einer Zeile und enthält die IP-

Adresse, gefolgt von dem zugeordneten Namen. Die Mac OS X 10.4-Hosts-Datei enthält keine Einträge für Rechner, sondern für besondere Adressen: der Loopback-Adresse 127.0.0.1 wird der Name localhost und der globalen Broadcast-Adresse 255.255.255.255 der Name broadcasthost zugewiesen. Der letzte Eintrag weist der IPv6-Loopback-Adresse ebenfalls den Namen localhost zu.

Außerhalb der Startphase wird normalerweise das datenbankgestützte Verfahren DNS verwendet. Domain Name System (DNS) wird heutzutage von nahezu allen TCP/IP-Hosts verwendet, da dieses im Gegensatz zum tabellengestützten Verfahren ohne weiteres Millionen von Namen verwalten kann und für automatische Verteilung von neuen Einträgen sorgt.

DNS basiert auf einem hierarchisch aufgebauten Namensraum. Dieser Namensraum wird dabei nicht etwa in einer zentralen Datenbank verwaltet, sondern von Tausenden von DNS-Servern, die jeweils nur für einen Teil des Namensraums zuständig sind.

Die Namen aus diesem Namensraum ähneln in ihrem Aufbau Pfadbezeichnungen in einem hierarchischen Dateisystem. Statt des in Pfaden oft verwendeten Schrägstrichs (oder Doppelpunkts wie im klassischen Mac OS) werden bei DNS-Namen jedoch zur Abgrenzung der Namensbestandteile Punkte verwendet. Gleichzeitig werden die einzelnen Namensbestandteile, anders als bei Pfaden, nicht von links nach rechts, sondern von rechts nach links geschrieben.

Der DNS-Name der Apple Website *www.apple.com.* (der Name endet mit einem Punkt) besteht demnach aus drei Namensbestandteilen. Der Punkt am Ende des DNS-Namens wird von vielen Anwendungsprogrammen als ein absoluter DNS-Name interpretiert (ähnlich einem absoluten Pfad). DNS-Namen ohne diesen Punkt werden manchmal als relativ zu einem vorher festgelegten Standardnamensraum (z. B. dem Namensraum eines Unternehmens) interpretiert.

Die „Wurzel“ des DNS-Namensraums ist die so genannte *Root Domain*, welche von einer relativ kleinen Gruppe von 13 Root-Servern verwaltet und mit einem einzelnen Punkt bezeichnet wird. Direkt unterhalb der Root Domain befinden sich die so genannten Top-Level Domains (kurz TLD): die com-Domain ist eine solche TLD. Neben der com-Domain gibt es aber auch viele andere, wie beispielsweise die edu-Domain oder die deutsche de-Domain. Alle TLDs werden jeweils von eigenen Servern verwaltet.

Die Subdomains (also Unter-Domains) unterhalb der TLDs werden normalerweise wiederum von den Servern der jeweiligen Subdomain-Inhaber verwaltet – die Subdomain apple.com. wird also von Apple selbst verwaltet. Dem Subdomain-Inhaber steht es dabei frei, die komplette Subdomain von einem einzigen Server verwalten zu lassen, oder weitere Subdomains zu erstellen (z. B. euro.apple.com.) und diese von selbstständigen Servern verwalten zu lassen.

Auf diese Weise ist es nicht erforderlich, dass die Root-Server (oder irgend ein anderer DNS-Server) alle Zuordnungen selbst kennen. Ein der heutigen

Größe des Internets mit Millionen von Hosts entsprechender Datenbestand wäre auch gar nicht bewältigbar. Stattdessen kennt jeder DNS-Server immer nur einen Teil des DNS-Namesraums. Teile des eigenen DNS-Namensraums können an nachgelagerte DNS-Server ausgelagert werden und werden über entsprechende Einträge in der Datenbank des auslagernden DNS-Servers gekennzeichnet. Auf diese Weise lagern die Root-Server z. B. den gesamten *com*-Namensraum an eine Gruppe von für diese TLD zuständigen DNS-Servern aus. Diese wiederum haben keine Kenntnis über Hosts im *apple.com*-Namensraum, sondern enthalten nur einen Hinweis auf die für diesen Namensraum zuständigen DNS-Server von Apple.

Will man mit Hilfe des DNS-Systems nun herausfinden, zu welcher IP-Adresse der Name www.apple.com. (in der Praxis lässt man den letzten Punkt meistens weg) gehört, und kennt die IP-Adresse des eigentlich zuständigen DNS-Servers von Apple nicht, dann muss man zuerst die Root-Server befragen. Die Root-Server werden eine solche Anfrage nicht direkt beantworten, sondern auf den für die com-Domain zuständigen Server verweisen und dessen IP-Adresse mitteilen. Doch auch dieser Server wird keine direkte Antwort liefern, sondern lediglich die IP-Adresse des zuständigen Apple-DNS-Servers zurückgeben. Erst dieser Apple-DNS-Server wird, als der Zuständige für die apple.com.-Domain, die entsprechende IP-Adresse liefern.

Andersherum will man manchmal auch wissen, welcher DNS-Name zu einer bestimmten IP-Adresse gehört. Um solche Anfragen effizient, also ohne alle DNS-Server absuchen zu müssen, bearbeiten zu können, bedient man sich eines Kunstgriffs: man reserviert einen Teil des DNS-Namensraums für IP-Adressen und ordnet damit jeder IP-Adresse einen eigenen DNS-Namen zu.

Als Wurzel dient dabei der DNS-Namensraum *in-addr.arpa*. In diesem Namensraum gibt es 256 mit den Zahlen von 0 bis 255 benannte Unter-Domains: *0.in-addr.arpa*, *1.in-addr.arpa*, usw. bis *255.in-addr.arpa*. Diese erste Ebene entspricht dem ersten Byte einer IP-Adresse bzw. der von links gesehen ersten Zahl in der üblichen Dezimal-Notation. Jede dieser Unter-Domains hat wiederum 256 Unter-Domains, die dem zweiten Byte einer IP-Adresse bzw. der zweiten Zahl von links entsprechen. Ebenso hat jede dieser Unterdomains auch wieder 256 Unter-Domains, die dem dritten Byte und diese wiederum 256 Unter-Domains, die dem vierten Byte entsprechen. Diese Domains auf der untersten Hierarchiestufe enthalten dann Einträge, die auf die DNS-Namen der Hosts mit den entsprechenden IP-Adressen verweisen.

Um also beispielsweise den zur IP-Adresse 17.254.0.91 passenden DNS-Namen zu erfahren, muss man nach dem DNS-Namen 91.0.254.17.in-addr.arpa suchen und findet dort einen Verweis (DNS-Datensatz vom Typ PTR) auf *www.apple.com*.

Die Verantwortung für die Bereitstellung und Pflege der DNS-Datensätze unterhalb der *in-addr.arpa*-Domain wird ebenso wie in allen anderen Fällen delegiert. Mit der Übernahme eines öffentlichen IP-Adressenbereichs wird

man normalerweise in die Pflicht genommen, auch den entsprechenden Teil der *in-addr.arpa*-Domain mit Hilfe eines DNS-Servers abzudecken.

Da ein zusammenhängender IP-Adressenbereich immer ein gemeinsames Präfix hat (z. B. gehören alle IP-Adressen 17.0.0.0 bis 17.255.255.255, also das A-Klasse-Netz 17, Apple Computer), werden die IP-Adressen als DNS-Namen in der *in-addr.arpa*-Domain quasi rückwärts geschrieben. Obwohl auf den ersten Blick gewöhnungsbedürftig, erlaubt diese Anordnung die einfache Delegation der DNS-Verantwortung für zusammenhängende IP-Bereiche: um beispielsweise die Verantwortung für das gesamte A-Klasse-Netzwerk 17 an Apple Computer zu delegieren, genügt ein einziger, die Unter-Domain *17.in-addr.arpa* betreffender Eintrag in der Datenbank des für die *in-addr.arpa* zuständigen DNS-Servers.

Damit ein Host DNS-Namen in IP-Adressen und IP-Adressen in DNS-Namen umwandeln kann, benötigt er einen so genannten DNS-Resolver. Ein DNS-Resolver schickt Anfragen an DNS-Server, interpretiert deren Antworten und reicht diese Antworten an Anwendungsprogramme.

Die meisten DNS-Resolver sind allerdings recht einfach und nicht in der Lage, Verweise auf andere DNS-Server zu interpretieren. Diese Aufgabe müssen lokale DNS-Server übernehmen – sie leiten die Anfragen der DNS-Resolver an die am besten geeigneten DNS-Server weiter. Dazu speichern lokale DNS-Server die Ergebnisse aller Anfragen für einen bestimmten, in den Antworten der zuständigen DNS-Server definierten Zeitraum, und können auf diese Weise viele Anfragen gleich direkt beantworten.

Diese lokalen DNS-Server können, müssen aber nicht, gleichzeitig auch eine Domain-Zuständigkeit haben. Tun sie das nicht, dann spricht man von reinen *Caching*-DNS-Servern. Diese Art von DNS-Servern ergänzt lediglich die DNS-Resolver der einzelnen Hosts und liefert DNS-Informationen ausschließlich aus zweiter Hand.

2.1.6 BOOTP und DHCP

Im Gegensatz zu einigen anderen Protokollfamilien wie etwa AppleTalk erfordert TCP/IP vor der eigentlichen Verwendung eine Reihe von Einstellungen, die an jedem Host im Netzwerk – normalerweise manuell – vorgenommen werden müssen.

Dieser Vorgang kann mit Hilfe von Konfigurationsservern automatisiert werden. Dabei sind BOOTP und DHCP die verbreitetsten Verfahren, die bei TCP/IP-Konfigurationsservern zur Anwendung kommen.

Das Bootstrap Protocol (BOOTP) ist das ältere und etwas einfachere Verfahren. Um die gültigen Konfigurationsparameter zu erhalten, sendet ein BOOTP-Client eine BOOTREQUEST genannte Anfrage mit Hilfe von UDP an die Broadcast-Adresse 255.255.255.255.

Der BOOTP-Server antwortet auf eine solche Anfrage mit einem BOOTREPLY-Paket, mit dessen Hilfe er dem Client eine breite Palette von Konfigurationsparametern übermitteln kann, darunter u. a. die IP-Adresse, die

Subnetzmaske, die Router-Adresse sowie die DNS-Server-Adresse. Interessant dabei ist, dass auch der Server für die Antwort die Broadcastadresse verwendet: das ist natürlich notwendig, weil der Client zu diesem Zeitpunkt seine eigene IP-Adresse noch nicht kennt und daher ein an ihn adressiertes Paket nicht erkennen würde.

Die Zuordnung von IP-Adressen zu Hosts geschieht bei Verwendung von BOOTP ausschließlich statisch. Der Netzwerkadministrator legt dabei exakt fest, welchem Host welche IP-Adresse zugeordnet werden soll. Um die Hosts für die Zuordnung der IP-Adresse eindeutig zu identifizieren, wird dazu einfach die eindeutige Hardware-Adresse der verwendeten Netzwerkschnittstelle verwendet.

Das Dynamic Host Configuration Protocol (DHCP) baut auf BOOTP auf. Genauso wie bei Verwendung von BOOTP tauschen DHCP-Clients und DHCP-Server UDP-Pakete vom Typ BOOTREQUEST und BOOTREPLY aus, um Konfigurationsparameter zu übermitteln. Dabei kann im DHCP-Verfahren eine noch breitere Palette an Konfigurationsparametern übermittelt werden.

In der Praxis ist jedoch wichtiger, dass im DHCP-Verfahren im Gegensatz zum BOOTP die IP-Adressen wahlweise auch automatisch zugeteilt werden können. Dazu muss der Netzwerkadministrator lediglich einen Pool von Adressen definieren, den der DHCP-Server zu verwalten hat. Erreicht den DHCP-Server nach der Definition eines solchen Pools eine BOOTREQUEST-Anfrage, dann teilt er dem Client einfach die nächste freie IP-Adresse aus seinem Pool zu.

Gleichzeitig ist es mit DHCP möglich, die Nutzungsdauer von IP-Adressen zeitlich zu begrenzen. In diesem Fall vergibt der DHCP-Server IP-Adressen nur auf eine vorher bestimmte Zeit. Ist die vereinbarte Zeit abgelaufen, muss die „gemietete“ IP-Adresse an den Server zurückgegeben oder eine Verlängerung der Mietdauer (engl. *DHCP lease*) beantragt werden. Wird die Mietdauer nicht verlängert (z. B. weil der Client abgeschaltet oder aus dem Netzwerk entfernt wurde), kehrt eine solche abgelaufene IP-Adresse automatisch in den Pool zurück und kann erneut – auf Zeit – vergeben werden.

Obwohl die automatische Adressenzuteilung in vielen Fällen sehr praktisch ist, findet sie ihre Grenzen dort, wo auf bestimmten Hosts bestimmte Dienste laufen, auf die Clients zugreifen müssen (Server). Da in klassischen TCP/IP-basierten Netzwerken Dienste ausschließlich über den Rechnernamen bzw. (mittelbar über DNS) über die IP-Adresse des entsprechenden Hosts gefunden werden können, sind wechselnde IP-Adressen für Server natürlich nicht wünschenswert.

2.1.7 Bonjour/ZeroConf

Mit AppleTalk war der Aufbau eines Netzwerks sehr einfach. Ausgestattet mit einem Ethernet-Hub oder Switch konnte der Anwender einfach mehrere Macintosh-Computer zusammenstecken und sofort arbeiten, ohne sich um die

Konfiguration der Netzwerkparameter zu kümmern, und das ohne BOOTP oder DHCP oder gar DNS-Server aufsetzen zu müssen.

Um in einem AppleTalk-Netzwerk auf einen Fileserver zugreifen zu können war es nicht erforderlich, den Namen oder gar die Netzwerk-Adresse des Server-Rechners zu kennen. Die Benutzer waren es gewohnt, dass der Rechner in der Lage war, selbstständig die im Netzwerk verfügbaren Fileserver zu finden und in einer Liste anzuzeigen. Die Verwendung von Verzeichnisdiensten und ähnlichen Hilfsmitteln zu diesem Zweck war nicht erforderlich.

In TCP/IP-Netzwerken war dieser Komfort lange Zeit nicht möglich. Geändert hat sich das erst mit der Implementierung von *Rendezvous* in Mac OS X 10.2. Rendezvous war die Apple-eigene Umsetzung des neuen Internet-Standards ZeroConf (bzw. Zero Configuration Networking), an dessen Entwicklung Apple maßgeblich beteiligt war. Seit der Einführung von Mac OS X 10.4 wurde Rendezvous aus rechtlichen Gründen in *Bonjour* umbenannt. Bonjour, Rendezvous und ZeroConf werden daher oft als Synonyme gebraucht – im Kontext der Mac-OS-X-Plattform spricht man meistens von Bonjour bzw. Rendezvous, im Kontext anderer Plattformen (Windows, Linux u. a.) meistens von ZeroConf.

Um sich dem Komfort von AppleTalk anzunähern, mussten für Bonjour TCP/IP-konforme Lösungen für drei Teilprobleme gefunden werden:

- die Zuordnung von IP-Adressen zu Hosts ohne die Verwendung eines BOOTP- oder DHCP-Servers,
- die Zuordnung von Namen zu IP-Adressen ohne die Verwendung eines DNS-Servers,
- das Auffinden von Serverdiensten ohne vorherige Kenntnis des Servernamens bzw. der IP-Adresse des Servers.

Bonjour schlägt für jedes dieser Teilprobleme an AppleTalk angelehnte, aber mit TCP/IP vereinbare Lösungen als unabhängige Teilkomponenten vor. In einem vom restlichen Internet völlig getrennten Netzwerk ohne vorhandene Infrastruktur (DHCP- und/oder DNS-Server) werden alle drei Teilkomponenten von Bonjour benötigt. Eine solche Situation entsteht beispielsweise regelmäßig, wenn mehrere Spieler zusammenkommen, um ein Mehrbenutzer-Spiel zu spielen, und dazu eine Ad-hoc-Vernetzung aufbauen müssen.

Bonjour ist aber auch dann sinnvoll, wenn bereits eine gewisse Infrastruktur existiert. Gerade im privaten Bereich, aber auch in kleineren Unternehmen werden für den Internetzugang häufig kompakte Router eingesetzt, die u. a. auch einen DHCP-Server enthalten. In einer solchen Situation übernimmt normalerweise der Router die Zuordnung von IP-Adressen zu Hosts, ein DNS-Server ist aber meistens nicht vorhanden.

Aber selbst in größeren Netzwerken, die sowohl über einen DHCP-Server, als auch über einen DNS-Server verfügen, kann Bonjour sinnvoll eingesetzt werden. In solchen Netzwerken kann das Bonjour-Verfahren zum Auffinden von Serverdiensten eine sinnvolle Ergänzung und eine große Arbeitserleichterung für die Anwender darstellen.

Zuordnung von IP-Adressen zu Hosts

Die Bonjour-eigene automatische Zuordnung von IP-Adressen tritt immer nur dann in Aktion, wenn die Netzwerkschnittstelle nicht auf eine andere Weise konfiguriert wurde (also die IP-Adresse weder manuell noch automatisch, z. B. mit Hilfe eines DHCP-Servers, zugeordnet wurde).

Beim Bonjour-Verfahren ordnet sich jeder Host selbstständig die IP-Adresse zu. Dazu wählt er aus einem dafür vorgesehenen Bereich (169.254.1.0 bis 169.254.254.255) zufällig eine Adresse aus und konfiguriert seine Netzwerkschnittstelle entsprechend.

Da die auf diese Weise zugeordneten Adressen natürlich nicht unbedingt eindeutig sind, muss vor der ersten Benutzung überprüft werden, ob die ausgewählte Adresse nicht vielleicht doch schon von einem anderen Host verwendet wird.

In Ethernet-Netzwerken kann dies auf einfache Weise mit Hilfe einer ARP-Anfrage geschehen, in der nach dem Eigentümer der gewählten IP-Adresse gefragt wird. Kommt auch nach Wiederholung der Anfrage nach einer festgelegten Zeit keine Antwort, kann man davon ausgehen, dass die Adresse noch frei ist. Antwortet hingegen ein anderer Host auf eine dieser Anfragen, muss man eine andere IP-Adresse aus dem vorgesehenen Bereich wählen und den Überprüfungsprozess mit Hilfe von ARP wiederholen – so lange, bis eine freie IP-Adresse gefunden worden ist.

In der Praxis müssen natürlich noch zahlreiche Sonder- und Fehlerfälle beachtet werden – das Grundprinzip der Zuordnung folgt jedoch diesem einfachen Schema. Für den Anwender bedeutet das, dass Bonjour-fähige Rechner in der Lage sind, sich völlig selbstständig unterschiedliche IP-Adressen zuzuordnen und damit ohne Benutzereingriff die Grundvoraussetzungen für IP-basierte Kommunikation geschaffen werden.

Zuordnung von Namen zu IP-Adressen

IP-Adressen alleine sind natürlich alles andere als benutzerfreundlich. Ebenso ist der Aufwand, einen eigenen DNS-Server aufzusetzen, einen eigenen Namensraum zu definieren und laufend zu pflegen, nicht unerheblich. Aus diesem Grund beinhaltet Bonjour auch ein Verfahren zur Zuordnung von Namen zu IP-Adressen ohne dedizierte DNS-Server: Multicast DNS (kurz mDNS).

Das mDNS-Verfahren ähnelt dem DNS-Verfahren, kommt jedoch ohne herkömmliche (unicast) DNS-Server aus. Jeder Bonjour-fähige Host verfügt stattdessen über einen eigenen mDNS-Server, den so genannten *mDNS-Responder*. Dieser kennt den vom Benutzer oder vom Hersteller festgelegten Bonjour-Namen des Hosts und beantwortet Anfragen nach diesem Namen mit der hosteigenen IP-Adresse.

Der Bonjour-Name eines Mac-OS-X-Hosts wird in den Systemeinstellungen *Sharing* festgelegt (siehe Abb. 2.5). Aus Vereinfachungsgründen wird in

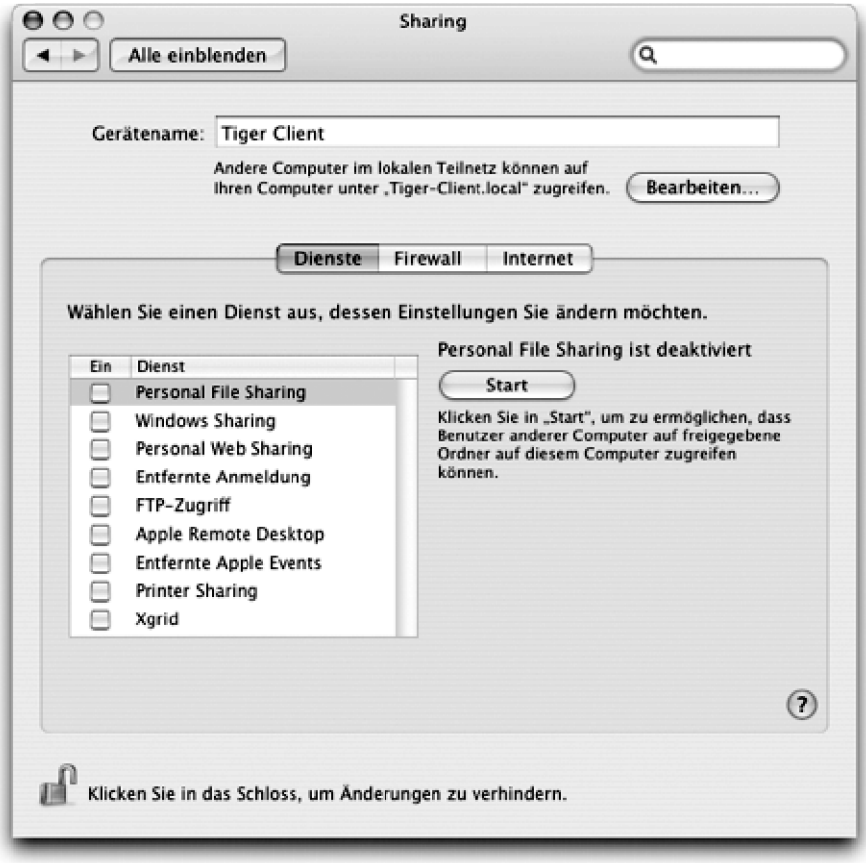


Abb. 2.5. Über Systemeinstellungen Sharing kann der Anwender den Bonjour-Namen seines Rechners festlegen. Hier wurde der Name „PowerMac.local“ automatisch aus dem AppleTalk-Namen „PowerMac“ abgeleitet.

Mac OS 10.4 der Bonjour-Name aus dem vom Benutzer konfigurierbaren AppleTalk-Namen abgeleitet. Falls dabei im AppleTalk-Namen Zeichen vorkommen, die in Bonjour-Namen nicht erlaubt sind, werden sie automatisch durch einen Bindestrich ersetzt. Über einen separaten, mit Hilfe der Schaltfläche „Bearbeiten“ aufzurufenden Dialog kann der Anwender aber ebenso einen vom AppleTalk-Namen gänzlich abweichenden Bonjour-Namen definieren.

Alternativ können AppleTalk-Name und der Bonjour-Name unabhängig voneinander auch mit dem Befehl *scutil* geändert werden:

```
$ sudo scutil --set ComputerName "AppleTalk Name"
$ scutil --get ComputerName
AppleTalk Name
$ sudo scutil --set LocalHostName Bonjour-Name
$ scutil --get LocalHostName
Bonjour-Name
```

Um Bonjour-Namen von anderen DNS-Namen abzugrenzen wurde für Bonjour eine eigene Top-Level Domain definiert: *.local.*. Namen, die mit dieser TLD enden, werden nicht über einen DNS-Server aufgelöst, sondern als Multicast (mDNS verwendet die Multicast-Adresse 224.0.0.251) an alle mDNS-Responder im lokalen Netzwerk geschickt.

Da zudem mDNS mit Port 5353 und herkömmliches DNS mit Port 53 auch unterschiedliche Ports verwenden, können beide Verfahren ohne Weiteres koexistieren. Ein Host kann neben einem „offiziellen" DNS-Namen (z. B. www.apple.com) auch über einen lokalen mDNS-Namen (z. B. unserwebserver.local) verfügen. Das mDNS-Verfahren funktioniert außerdem völlig unabhängig davon, ob IP-Adressen automatisch im oben präsentierten Bonjour-Verfahren oder auf herkömmliche Art und Weise zugeteilt worden sind.

Suche nach Serverdiensten

Der mDNS-Responder kann aber nicht nur Adressanfragen (DNS Resource Record Typ A) beantworten, sondern versteht darüber hinaus auch so genannte *Pointer*- (DNS Resource Record Typ PTR), *Service*- (DNS Resource Record Typ SRV) und *Text*-Anfragen (DNS Resource Record Typ TXT). Mit Hilfe dieser DNS-Standardanfragen sowie einiger Konventionen zu deren Verwendung ist es möglich, im lokalen Netzwerk angebotene Serverdienste automatisch zu finden.

Ein Serverdienst, der von Benutzern automatisch gefunden werden soll, muss sich bei jedem Start bei einem mDNS-Responder registrieren. Normalerweise erfolgt diese Registrierung natürlich bei dem mDNS-Responder, der auf dem gleichen Host läuft wie der Serverdienst selbst.

Für die Registrierung werden mindestens der Name des Dienstes, sein Typ sowie der (TCP oder UDP) Port benötigt, über den der Dienst erreichbar ist. Bei der Registrierung legt der mDNS-Responder in seiner internen Datenbank zwei Einträge an: einen Service-Eintrag (SRV) und einen Pointer-Eintrag (PTR).

Der Service-Eintrag legt dabei den Host-Namen und die Port-Nummer fest, unter der ein bestimmter Dienst erreichbar ist. Ist dem Benutzer (bzw. dem Client-Programm) der Service-Name bereits bekannt, kann mit Hilfe einer DNS-Standardanfrage vom Typ SRV der dazugehörige Host-Name und die Port-Nummer bezogen werden.

Der Pointer-Eintrag weist dem Dienst einen bestimmten Diensttyp zu. Auf diese Weise kann mit Hilfe einer DNS-Standardanfrage vom Typ PTR eine Liste aller Dienste eines bestimmten Typs generiert werden. Ein Web-Browser kann so z. B. die Liste aller Web-Server anfordern.

Optional können noch Text-Einträge für zusätzliche Informationen angelegt werden. Auch hierbei handelt es sich um Einträge, die mit einer DNS-Standardanfrage (vom Typ TXT) ausgelesen werden können. In Bonjour werden Text-Einträge als Namen/Werte-Paare interpretiert, welche die Anwendungsentwickler für Status-Informationen u.ä. verwenden können.

Im Gegensatz zu den weiter oben beschriebenen Verfahren zur automatischen Vergabe von IP-Adressen und zur Zuordnung von lokalen Namen zu IP-Adressen, ist das Bonjour-Verfahren zur Auffindung von Serverdiensten nicht grundsätzlich nur auf lokale Netzwerke beschränkt.

Da in diesem Verfahren lediglich bestimmte Konventionen für die Verwendung von DNS-Standardeinträgen bzw. -anfragen verwendet werden, aber ansonsten keine Protokolländerungen spezifiziert wurden, könnten neben lokalen mDNS-Respondern auch herkömmliche DNS-Server zur Registrierung verwendet werden.

In den bisherigen Implementierungen von Bonjour (also auch in Mac OS X 10.4) fehlen derzeit noch die entsprechenden Schnittstellen, um einen Dienst für eine beliebige DNS Domain zu registrieren oder nach Diensten außerhalb der .local TLD zu suchen. Grundsätzlich wäre das protokollseitig aber möglich und würde die Suche nach Serverdiensten auch außerhalb des lokalen Netzwerks vereinfachen.

2.1.8 IPv6

IPv6 ist als Nachfolger des im Abschn. 2.1 vorgestellten Internet-Protokolls IP konzipiert worden und soll es langfristig ersetzen. Die Ziffer „6“ steht dabei für die Versionsnummer des Protokolls – bei der derzeit noch aktuellen Version des Internet-Protokolls handelt es sich um die Version 4 (kurz IPv4).

IPv6 ist der direkte Nachfolger von IPv4, ohne dass es ein IPv5 je gegeben hätte. Die Versionsnummer 5 musste aus technischen Gründen übersprungen werden, da sie für ein anderes Protokoll – SP – schon vergeben worden war.

SP bzw. das Internet Stream Protocol wurde in den 70er Jahren für die Übertragung von Sprach- und Bildinformationen entwickelt, und ist heute nur noch wenig bekannt. Da dieses Protokoll mit IPv4 koexistieren sollte, musste ein Weg gefunden werden, um SP-Pakete und IP-Pakete unterscheiden zu können. Man hat sich dazu entschlossen, hierfür einfach das Versionsfeld des IP Headers zu verwenden (die ersten vier Bits) und definierte die Nummer 5 als die Versionsnummer von SP.

Den Entwicklern des IP-Nachfolgers blieb damit nichts anderes übrig, als die nächste freie Versionsnummer zu wählen: die Nummer 6.

Eines der wichtigsten Entwurfsziele für IPv6 war die Vergrößerung des für Hosts verfügbaren Adressraums. IPv4 wurde 1981 im RFC 791 definiert und ab 1983 im ARPAnet (dem Vorläufer des heutigen Internets) exklusiv verwendet. Während in dieser Anfangsphase nur einige wenige Hundert Hosts über IP-Adressen verfügten, wuchs deren Zahl bis zum Jahr 1989 auf über 100.000. In den Neunzigern wurde das Internet zu einem Massenphänomen und entsprechend stieg auch die Anzahl der Hosts auf ca. 100 Millionen.

Bereits Anfang der 90er Jahre glaubte man, dass der Bedarf an IP-Adressen schon bald die Möglichkeiten von IPv4 übersteigen wird. Es schien nur noch eine Frage der Zeit zu sein, dass die IP-Adressen eines Tages ausgehen.

Vor diesem Hintergrund wurde mit IPv6 ein Nachfolgerprotokoll konzipiert, welches neben anderen Detailverbesserungen vor allem einen deutlich größeren Adressraum bietet. Während IPv4 32 Bits für die Kennzeichnung von Host-Adressen verwendet und man damit 4.294.967.296 (also etwas mehr als vier Milliarden) Hosts eindeutig identifizieren kann, wurden mit IPv6 erstmals 128-Bit-Adressen eingeführt, so dass theoretisch die unvorstellbar große Zahl von ca. $3{,}4 * 10^{38}$ Hosts eindeutig identifiziert werden könnte.

Wegen ihrer Länge werden IPv6-Adressen nicht mehr wie IPv4-Adressen dezimal (z. B. 17.112.152.32), sondern hexadezimal geschrieben. Dabei werden 16-Bit-Gruppen an Stelle von 8-Bit-Gruppen gebildet und durch Doppelpunkte an Stelle von Punkten getrennt, z. B.:

```
fe80:0000:0000:0000:020a:95ff:fef4:bde1
```

Zur Vereinfachung kann eine Folge von 16-Bit-Gruppen mit Wert 0 weggelassen werden, also z. B.

```
fe80::020a:95ff:fef4:bde1
```

statt

```
fe80:0000:0000:0000:020a:95ff:fef4:bde1
```

Die ersten 64 Bit einer IPv6-Adresse werden normalerweise als Netzwerkadresse, die restlichen 64 Bit als Hostadresse interpretiert – diese Bitanzahl ist jedoch, ähnlich wie bei einer IPv4-Adresse, konfigurierbar.

Bestimmte IPv6 sind für besondere Zwecke reserviert. Interessant sind u. a.:

- ::0 ist die „undefinierte“ Adresse (analog 0.0.0.0)
- ::1 ist die Loopback-Adresse (analog 127.0.0.1)
- Adressen mit Präfix fe80 bis febf (siehe Beispiel weiter oben) sind lokale, für die automatische Vergabe vorgesehene Adressen (analog dem Adressbereich 169.254.1.0 bis 169.254.254.255)
- Adressen mit Präfix fec0 bis feff sind zur privaten Verwendung vorgesehene Adressen (analog den Netzwerken 10.0.0.0, 172.16.0.0 bis 172.31.0.0 und 192.168.0.0 bis 192.168.255.0).

IPv6 hat derzeit bei uns noch wenig praktische Bedeutung. Die prognostizierte IP-Adressenknappheit ist in der befürchteten Form noch nicht eingetreten – nicht zuletzt durch die weit verbreitete Nutzung von privaten IPv4-Adressen und die Verwendung von NAT. Mit NAT verschwinden häufig selbst sehr große Netzwerke hinter einer einzigen öffentlichen IP-Adresse.

In anderen Teilen der Welt, vor allem in Ländern, die erst verhältnismäßig spät den Anschluss an das Internet vollzogen haben, ist die Lage jedoch deutlich angespannter. Es ist zudem damit zu rechnen, dass sich die Adressenknappheit in Zukunft aufgrund von technischen Entwicklungen deutlich verschärfen wird.

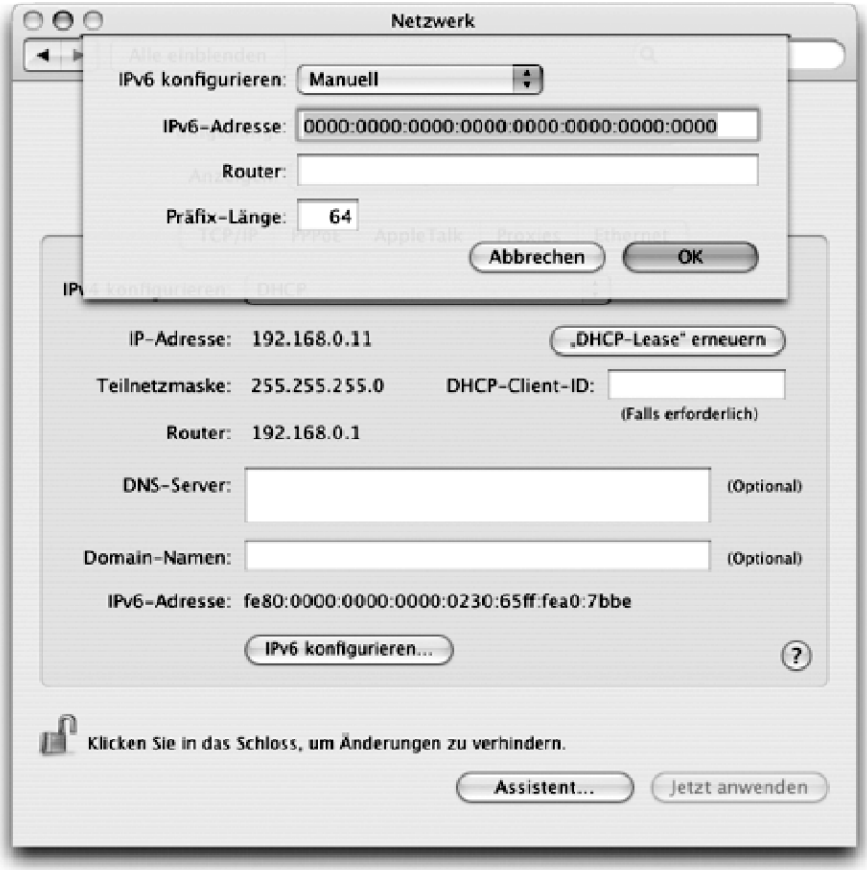


Abb. 2.6. Die IPv6-Adresse und die Routeradresse lassen sich seit Mac OS X 10.3 auch manuell eingegeben.

Mac OS X ist seit der Version 10.2 mit Unterstützung für IPv6 ausgestattet. Seit der Version 10.3 kann über die Systemeinstellung Netzwerk auch IPv6 konfiguriert werden. Über ein einfaches Sheet kann man dort zwischen der automatischen und manuellen Konfiguration umschalten, oder wahlweise die IPv6-Unterstützung für eine Netzwerkschnittstelle ganz abschalten.

Bei der manuellen Konfiguration können die IPv6-Adresse und die Routeradresse manuell festgelegt werden. Ebenso kann man mit der Präfix-Länge angeben, in welchem Verhältnis die Adressen-Bits zwischen Netzwerk- und Host-Adresse aufgeteilt werden sollen.

Alternativ kann man auch den Befehl *ip6* verwenden, um die IPv6-Unterstützung für bestimmte (oder auch alle) Netzwerkschnittstellen zu aktivieren oder zu deaktivieren:

```
$ ip6
Usage:
       Start up IPv6 on ALL interfaces:      -a
       Shut down IPv6 on ALL interfaces:     -x
       Start up IPv6 on given interface:     -u [interface]
       Shut down IPv6 on given interface:    -d [interface].
```

2.2 Die AppleTalk-Protokollfamilie

Ursprünglich wurde AppleTalk als eine Technologie geplant, die lediglich den parallelen Zugriff von mehreren Macintosh-Anwendern auf einen (damals sehr teuren) LaserWriter ermöglichen sollte. Dennoch wurde AppleTalk von Apple von Anfang an als ein „richtiges“ Netzwerkprotokoll konzipiert und als eine Familie von aufeinander aufbauenden Teilprotokollen entwickelt.

AppleTalk ist dementsprechend genauso wie TCP/IP grundsätzlich vom Übertragungsmedium unabhängig und unterstützt die unterschiedlichsten Verkabelungssysteme. Von größter praktischer Bedeutung sind heutzutage natürlich kabelgebundene und drahtlose Ethernet-Netzwerke nach IEEE 802.3 und 802.11. In der Vergangenheit wurde AppleTalk jedoch z. B. sehr häufig zusammen mit dem von Apple entwickelten proprietären Verkabelungssystem *LocalTalk* bzw. dem verwandten *PhoneNET*-System von Farallon Computing verwendet.

AppleTalk wurde nicht nur auf Apple-Macintosh-Systemen implementiert. Mit den *Windows 2000 Services For Macintosh* liefert Microsoft eine Implementierung von AppleTalk zusammen mit Windows 2000 Server Edition aus, die neben dem Protokoll einige Anwendungen wie einen AFP-Fileserver und einen AppleTalk-Router beinhaltet. Für Unix-basierte Systeme stehen mit dem älteren CAP (Columbia AppleTalk Package) und dem neueren Netatalk zwei freie Implementierungen zur Verfügung, die u. a. auch einen AFP-Fileserver und einen AppleTalk-Router beinhalten.

2.2.1 Einordnung in ein Fünf-Schichten-Modell

Ähnlich wie die TCP/IP-Protokolle lassen sich auch die zur AppleTalk-Familie gehörenden Protokolle in einem Fünf-Schichten-Modell darstellen:

1. Die *Physikalische Schicht* stellt die grundsätzliche Fähigkeit her, Datenpakete über ein bestimmtes Übertragungsmedium von einem Rechner zu anderen zu versenden. Anders als TCP/IP beinhaltet AppleTalk mit LocalTalk die Spezifikation eines solchen Übertragungsmediums. Andere Übertragungsmedien wie z. B. IEEE-802.3- und 803.11-konforme Ethernets können aber genauso als Grundlage für AppleTalk verwendet werden wie das von Apple definierte LocalTalk.
2. Die *Verbindungsschicht* ermöglicht die Nutzung von bestimmten Übertragungsmedien. Apple hat mit dem *LocalTalk Link Access Protocol* (kurz

LLAP), dem *EtherTalk Link Access Protocol* (kurz ELAP) und dem TokenTalk Link Access Protocol (kurz TLAP) die Verwendbarkeit von AppleTalk in LocalTalk, Ethernet und Token-Ring-Netzwerken sichergestellt.

3. Die *Netzwerkschicht* nutzt die Verbindungsschicht, um den Versand von Datenpaketen zwischen beliebigen Rechnern zu ermöglichen. Die Netzwerkschicht wird in AppleTalk durch das *Datagram Delivery Protocol* (kurz DDP) realisiert. Mit Hilfe des *AppleTalk Address Resolution Protocols* (kurz AARP) lassen sich DDP-Adressen auf Adressen der Verbindungsschicht abbilden.
4. Die *Transportschicht* nutzt die Netzwerkschicht, um Anwendungen den Versand von Daten zu ermöglichen. AppleTalk stellt mit dem *AppleTalk Transaction Protocol* (kurz ATP), dem *Printer Access Protocol* (kurz PAP)[7], dem *AppleTalk Session Protocol* und dem *AppleTalk Data Stream Protocol* (kurz ADSP) eine sehr breite Auswahl an Transportprotokollen für unterschiedlichste Anwendungen zur Verfügung.
5. Die *Anwendungsschicht* besteht schließlich aus den Anwendungen, die auf den Diensten der Transportschicht aufsetzen. Das *AppleTalk Filing Protocol* kann z. B. ASP nutzen, um den Zugriff auf Dateien über Rechnergrenzen hinweg zu realisieren.

2.2.2 Datagram Delivery Protocol

Das *Datagram Delivery Protocol* (kurz DDP) ermöglicht ähnlich wie IP die Datenübertragung in einem Netzwerkverbund. Zur eindeutigen Identifikation von AppleTalk-Hosts in einem solchen Netzwerkverbund kommen in DDP 24 Bits lange Hostadressen zur Anwendung.

Die insgesamt 24 Bits lange DDP-Hostadresse besteht immer aus einer 16 Bits langen Netzwerkadresse und einer 8 Bits langen Knotenadresse (engl. Node ID). Die Netzwerkadresse identifiziert eindeutig ein bestimmtes lokales Netzwerk in einem Netzwerkverbund. Die Knotenadresse identifiziert eindeutig einen bestimmten Host innerhalb eines bestimmten Netzwerks, wobei die Knotenadresse 0 und die Knotenadresse 255 reserviert sind: die Knotenadresse 0 symbolisiert einen beliebigen Router und die Knotenadresse alle Hosts im Netzwerk (Broadcast).

Damit stehen pro Netzwerk 254 Knotenadressen zur Verfügung. Um Netzwerke mit mehr als 254 Macintosh-Rechnern aufbauen zu können, kann man einem lokalen Netzwerk auch mehrere Netzwerkadressen zuordnen. Theoretisch sind damit AppleTalk-Netzwerke mit mehr als 16 Millionen Hosts möglich[8].

[7] Trotz seines Namens ist PAP in seiner Anwendung nicht grundsätzlich auf die Ansteuerung von Druckern beschränkt.

[8] Die Zuordnung von mehreren Netzwerkadressen zu einem lokalen Netzwerk war in der ursprünglichen AppleTalk-Spezifikation nicht vorgesehen. Erst das 1989 vorgestellte *AppleTalk Phase 2* hob die Beschränkung von AppleTalk auf 254 Hosts pro Netzwerk auf.

Obwohl die DDP-Hostadresse in Mac OS X auch manuell festgelegt werden kann, ist die manuelle Konfiguration, anders als bei IP, ein seltener Sonderfall. Normalerweise wird die DDP-Adresse völlig automatisch, für den Benutzer unsichtbar, beim Systemstart bzw. beim Start der AppleTalk-Netzwerksoftware festgelegt.

Jeder AppleTalk-Host weist sich dazu beim Systemstart eine vorläufige DDP-Adresse selbst zu. Anschließend versucht er einen AppleTalk-Router zu kontaktieren, um eine „offizielle“ DDP-Adresse zu beantragen. Sollte kein Router erreichbar sein, wird die vorläufige DDP-Adresse einfach beibehalten. In AppleTalk-Netzwerken übernehmen Router demnach auch die Verteilung von Konfigurationsdaten – separate Dienste hierzu nach dem Vorbild von BOOTP oder DHCP sind nicht erforderlich.

Die AppleTalk-Netzwerkadressen 65280 bis 65534 sind für solche selbstzugewiesenen, „vorläufigen“ DDP-Adressen reserviert. Ein startender AppleTalk-Host wählt demnach zufällig eine dieser 254 Netzwerkadressen aus und ergänzt sie mit einer beliebigen Knotennummer aus dem Bereich 1 bis 253 (zusätzlich zu Knotennummer 0 und 255 ist in Netzwerken mit mehr als einer Netzwerkadresse auch die Knotennummer 254 reserviert und darf nicht verwendet werden).

Anschließend prüft er mit Hilfe des AppleTalk Address Resolution Protocols (kurz AARP), ob die Adresse bereits in Verwendung ist. Sollte das der Fall sein, muss eine andere Adresse gewählt werden. Insgesamt können auf diese Weise theoretisch also über 64.000 AppleTalk-Hosts ohne zentrale Administration betrieben werden.

In der Praxis erfordern Netzwerke dieser Größe natürlich regelmäßig den Einsatz eines AppleTalk-Routers zur Vergabe von DDP-Adressen, da der Aufwand zur Auffindung einer freien DDP-Adresse mit zunehmender Zahl von Hosts stark ansteigt. Ein solcher Router verfügt in der Regel über mehrere Netzwerkschnittstellen und verbindet damit mehrere unabhängige AppleTalk-Netzwerke miteinander. Jede Netzwerkschnittstelle eines AppleTalk-Routers kann als ein so genannter Seed-Port definiert werden. Dazu wird der Netzwerkschnittstelle ein Bereich von Netzwerkadressen zur Verteilung an die angeschlossenen Hosts zugewiesen. In einem Netzwerkverbund muss der Administrator des Routers sicherstellen, dass die zugewiesenen Netzwerkadressen eindeutig sind.

Um mehrere Netzwerkanwendungen auf einem Host unterscheiden zu können, verwendet DDP so genannte *Sockets*. Sockets werden mit Hilfe von 8 Bits langen Socket-Nummern eindeutig identifiziert. Damit stehen je Host insgesamt 254 Sockets zur Verfügung (die Socket-Nummern 0 und 255 sind reserviert und können nicht als Socket-Identifikatoren verwendet werden).

Man unterscheidet dabei zwischen statischen Sockets (engl. *Statically Assigned Sockets*, kurz SAS) mit Socket-Nummern von 1 bis 127 und dynamischen Sockets (engl. *Dynamically Assigned Sockets*, kurz DAS) mit Socket-Nummern von 128 bis 254. Die Hälfte des statischen Socket-Bereichs – mit

Socket-Nummern von 1 bis 63 – ist von Apple für AppleTalk-Hilfsprotokolle[9] reserviert und darf nicht verwendet werden. Die andere Hälfte – mit Socket-Nummern von 64 bis 127 – ist für experimentelle Zwecke reserviert. Da es, anders als bei UDP- und TCP-Port-Nummern, keine zentrale Registrierungsstelle für statische Socket-Nummern gibt, dürfen statische Sockets nicht in Software-Produkten verwendet werden.

2.2.3 Name Binding Protocol

Für die Benutzer ist die direkte Verwendung von DDP-Adressen und Socket-Nummern grundsätzlich nicht praktikabel. Nicht nur, dass Nummern an sich schwieriger zu merken sind als Namen: hinzu kommt noch, dass sowohl DDP-Adressen als auch Socket-Nummern in der Regel dynamisch vergeben werden, dem Benutzer a priori unbekannt sind und sich im Laufe der Zeit auch noch ändern können.

Um dieses Problem zu umgehen, verwendet AppleTalk das *Name Binding Protocol* (kurz NBP), um DDP-Adressen und Socket-Nummern auf Namen abzubilden. Die Kombination einer DDP-Adresse und einer Socket-Nummer wird in Zusammenhang mit NBP als *Network-visible Entity* (kurz NVE) bezeichnet – in der Regel handelt es sich bei einer NVE einfach um eine Serveranwendung, oder den Teil einer Serveranwendung.

Mit Hilfe von NBP lassen sich NVEs Namen zuweisen. Ein NVE-Name ist eine maximal 98 Zeichen lange Zeichenkette, die aus einem vom Benutzer gewählten Objektnamen, einer Typbezeichnung und einer Zonenbezeichnung besteht. Jeder dieser Bestandteile kann aus maximal 32 Zeichen bestehen. Als Trennzeichen zwischen dem Objektnamen und der Typbezeichnung wird ein Doppelpunkt und als Trennzeichen zwischen Typbezeichnung und Zonenbezeichnung ein At-Zeichen (`@`) verwendet:

```
MeinServer:AFPServer@MeineAbteilung
```

In einem einfachen AppleTalk-Netzwerk ohne Router gibt es keine Zonen bzw. eine einzige Standardzone, die mit einem Stern (*) bezeichnet wird. In einem größeren Netzwerkverbund kann der Administrator mit Hilfe von AppleTalk-Routern Zonen definieren und diese den angeschlossenen Netzwerken zuweisen. Sind einem (Teil-)Netzwerk mehrere Zonen zugewiesen, kann der Benutzer angeben, zu welcher Zone ein an dieses Netzwerk angeschlossener AppleTalk-Host gehören soll. Nimmt der Benutzer keine Einstellung der Zone vor, gehört der Host automatisch zu der für das jeweilige Netzwerk definierten Standardzone.

Mit Hilfe von Zonen lassen sich auf diese Weise AppleTalk-Hosts relativ willkürlich, z. B. analog zur organisatorischen Zugehörigkeit der Hosts zu Abteilungen, zu Gruppen zusammenfassen.

[9] z. B. das sehr einfache AppleTalk Echo Protocol oder das weiter unten beschriebene Name Binding Protocol.

Um einen NVE-Namen in einer bestimmten Zone oder auch in der Standardzone zu registrieren, muss NBP überprüfen können, ob der gewünschte Name nicht etwa schon vergeben wurde. Der registrierende Host überprüft dabei immer zuerst, ob der Name nicht schon von einem anderen, eigenen Prozess in der lokalen NBP-Namen-Datenbank registriert wurde. Ist das der Fall, wird der Registrierungsversuch mit einer Fehlermeldung abgebrochen.

Ist der Name lokal noch frei, muss NBP sicherstellen, dass der zu registrierende Name noch nicht von einem anderen AppleTalk-Host registriert wurde. In einem Netzwerk ohne Router sendet NBP dazu ein spezielles NBP (Lookup)-Paket an alle Hosts im lokalen Netzwerk als DDP-Broadcast. Die Empfänger überprüfen daraufhin ihre eigene NBP-Namen-Datenbank und senden ggf. eine Antwort mit der DDP-Adresse und Socket-Nummer an den Initiator des Broadcasts. Da DDP-Pakete unterwegs verloren gehen können, wird der Broadcast zur Sicherheit mehrmals wiederholt.

In einem Netzwerk mit Routern und mehreren Zonen sendet NBP ein spezielles (Broadcast-Request-) Paket an den nächstgelegenen Router, um einen an eine bestimmte Zone gerichteten Broadcast auszulösen. Falls sich Zonen-Mitglieder in einem der an diesen Router angeschlossenen Netzwerke befinden, führt der Router stellvertretend für den registrierenden Host den Broadcast aus; gleichzeitig leitet er die Anfrage an alle anderen Router weiter, die Zonen-Mitglieder der betreffenden Zone verwalten.

3

Netzwerk – Anwendungen

3.1 Basiskonfiguration

Die Systemeinstellung *Netzwerk* (zugänglich über das Programm *Systemeinstellungen* aus dem Verzeichnis /Programme/Dienstprogramme) ermöglicht die Konfiguration aller wichtigen Netzwerkparameter (Abb. 3.1).

Mit dieser Systemeinstellung lassen sich die Netzwerkparameter für jede Netzwerkschnittstelle (z. B. Ethernet, AirPort, Bluetooth oder Modem) individuell festgelegen. Zusätzlich kann man auch den Status der aktivierten Netzwerkschnittstellen betrachten.

Die Benutzeroberfläche (Abb. 3.2) besteht im Wesentlichen aus den folgenden Komponenten:

- dem Pop-up-Menü „Umgebung“,
- dem Pop-up-Menü „Anzeigen“,
- dem eigentlichen Konfigurationsbereich,
- einem Schloss-Symbol,
- der Schaltfläche „Assistent. . . “
- und der Schaltfläche „Jetzt aktivieren“.

Mit Hilfe des Pop-up-Menüs „Umgebung“ kann man zwischen mehreren zuvor erstellten Konfigurationen umschalten. Über dieses Menü kann man außerdem neue Konfigurationen („Umgebungen“) anlegen und existierende Konfigurationen bearbeiten. Im Auslieferungszustand existiert genau eine Konfiguration: die Konfiguration „Automatisch“.

In dieser Standardkonfiguration sind alle gefundenen Netzwerkschnittstellen aktiviert (also z. B. die eingebaute Ethernet-Schnittstelle und die eventuell vorhandene AirPort-Karte für drahtloses LAN nach IEEE 803.11b oder 803.11g) und für den Bezug der TCP/IP-Netzwerkparameter über DHCP vorkonfiguriert. Die Konfiguration „Automatisch“ ist aber ansonsten eine

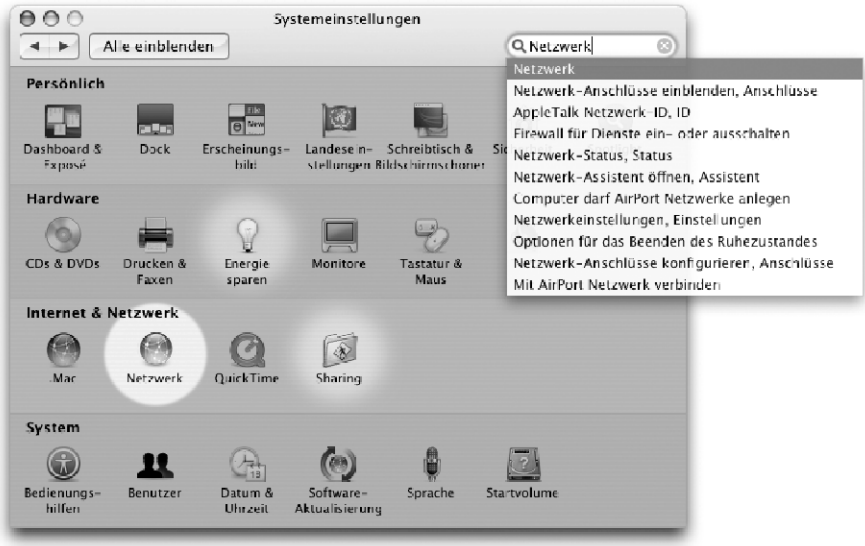


Abb. 3.1. Über die Systemeinstellung Netzwerk lassen sich alle wichtigen Netzwerkparameter konfigurieren. Weitere netzwerkbezogene Einstellungen findet man u. a. in den Systemeinstellungen Sharing und Energie sparen.

normale Konfiguration, so dass sie vom Anwender ohne weiteres umbenannt und auch geändert werden kann.

Alternativ kann man auch über das Apple-Menü (dort findet man ebenfalls ein Menü „Umgebungen“) oder mit Hilfe der Kommandozeile zwischen zuvor definierten Konfigurationen umschalten. In der Kommandozeile kann man dazu den Befehl `scselect` (system configuration select) verwenden:

```
$ scselect
Defined sets include: (* == current set)
   F1B99FFC-BFD6-45B7-B594-23FC61C47672 (Nur Ethernet)
 * 0    (Automatic)
$ scselect "Nur Ethernet"
CurrentSet updated to
   F1B99FFC-BFD6-45B7-B594-23FC61C47672 (Nur Ethernet)
$ scselect
Defined sets include: (* == current set)
 * F1B99FFC-BFD6-45B7-B594-23FC61C47672 (Nur Ethernet)
   0    (Automatic)
```

Als Argument akzeptiert `scselect` sowohl den eindeutigen Identifikator als auch den Namen einer Konfiguration. Ein Aufruf ohne Argumente erzeugt eine Liste aller definierten Konfigurationen.

Mit dem Pop-up-Menü „Zeigen“ kann man den Inhalt des Konfigurationsbereichs bestimmen. Man kann zwischen einer Status-Übersicht („Netzwerk-Status“), einer netzwerkschnittstellen-spezifischen Konfigurationsansicht (Auswahl jeweils über die Bezeichnung der Netzwerkschnittstelle) und einer allgemeinen Konfigurationsansicht („Netzwerk-Konfiguration“) umschalten.

Abb. 3.2. Mit der Systemeinstellung Netzwerk können alle wichtigen Netzwerkparameter auf einfache Weise konfiguriert werden.

Über die letztere kann man Netzwerkschnittstellen aktivieren und deaktivieren, deren Bezeichnungen ändern, Aliase anlegen und die Prioritätenfolge (mit Auswirkungen auf das Routing – siehe Abschn. 3.2) festlegen.

Wird ein Macintosh-Rechner also nach der Inbetriebnahme an ein Netzwerk mit einem konfigurierten DHCP-Server angeschlossen, fragt er automatisch während des Systemstarts nach einer freien IP-Adresse, der Teilnetzmaske, der IP-Adresse des Routers und der DNS-Server sowie nach eventuell vorgegebenen Domain-Namen für die Ergänzung von unvollständig angegebenen Rechnernamen (z. B. server statt server.example.com).

Sollte kein DHCP-Server antworten, wird Mac OS X der Schnittstelle unter der Annahme, dass der Rechner sich in einem rein lokalen Netzwerk ohne Verbindung nach außen befindet, automatisch eine IP-Adresse im Bonjour-Verfahren zuweisen.

Obwohl beim Bezug der Netzwerkparameter über DHCP meistens auch ohne Eingriffe des Benutzers eine funktionsfähige Netzwerkverbindung zu Stande kommt, besteht innerhalb der grafischen Benutzeroberfläche die Möglichkeit, einige zusätzliche Parameter manuell zu beeinflussen.

So kann man eine frei gewählte Zeichenkette als DHCP-Client-ID angeben. Normalerweise ist es nicht notwendig, explizit diese DHCP-Client-ID anzugeben. Ein DHCP-Client kann jederzeit auch ohne diese Angabe eindeutig anhand der Hardware-Adresse der verwendeten Schnittstelle vom DHCP-Server identifiziert werden. Kommt es jedoch zu einem Austausch der Netzwerkhardware, ändert sich auch diese Hardwareadresse und der DHCP-Client könnte so unter Umständen ungewollt eine andere, neue IP-Adresse bekommen. Um das zu verhindern, könnte man eine eindeutige DHCP-Client-ID, z. B. die Apple-Seriennummer, explizit angeben und den DHCP-Server dahingehend umkonfigurieren, dass eine solche Seriennummer und nicht die Hardwareadresse zur eindeutigen Identifikation des DHCP-Clients verwendet wird. In der Praxis wird die DHCP-Client-ID allerdings selten verwendet.

Ferner kann man auch zusätzliche Konfigurationsparameter für den DNS-Client, den so genannten Resolver, angeben. Der Resolver ist eine C-Bibliothek, welche von Kommandozeilentools und Anwendungsprogrammen dazu benutzt werden kann, DNS-Namen aufzulösen und die passenden IP-Adressen zurückzugeben. Obwohl sowohl IP-Adressen von DNS-Servern als auch ergänzende DNS-Namen über das DHCP-Protokoll empfangen werden können, besteht die Möglichkeit, weitere IP-Adressen und DNS-Namen hinzuzufügen.

Die über das DHCP-Protokoll empfangenen Parameter des Resolvers werden im Gegensatz zu den anderen Parametern leider nicht grafisch angezeigt. Stattdessen bleiben die betreffenden Felder immer leer. Die entsprechenden Einstellungen findet man aber jederzeit in der Datei /etc/resolv.conf[1] wieder. Der Versuchung, diese Datei manuell zu editieren, sollte man allerdings widerstehen, da sie von Mac OS X entsprechend der Systemeinstellung Netzwerk selbst verwaltet wird.

```
$ cat /etc/resolv.conf
nameserver 192.168.0.1
```

Eine vollständige Übersicht der vom DHCP-Server erhaltenen Parameter kann man mit Hilfe des Befehls `ipconfig` erhalten. Mit diesem Befehl lässt sich die zur Konfiguration verwendete Antwort des DHCP-Servers vollständig darstellen und darüber hinaus auch feststellen, welcher DHCP-Server (`server_identifier (ip)`) wirklich geantwortet hat:

```
$ ipconfig getpacket en0
op = BOOTREPLY
htype = 1
dp_flags = 0
hlen = 6
```

[1] /etc/resolv.conf ist ein symbolischer Link auf /var/run/resolv.conf

```
hops = 0
xid = 651061539
secs = 0
ciaddr = 0.0.0.0
yiaddr = 192.168.0.10
siaddr = 0.0.0.0
giaddr = 0.0.0.0
chaddr = 0:3:93:19:69:6a
sname = ?
file = ?
options:
Options count is 10
dhcp_message_type (uint8): ACK 0x5
server_identifier (ip): 192.168.0.1
subnet_mask (ip): 255.255.255.0
lease_time (uint32): 0x3840
option_overload (uint8): 0x3
router (ip_mult): {192.168.0.1}
domain_name_server (ip_mult): {192.168.0.1}
end (none):
end (none):
end (none):
```

Der Konfigurationsbereich der Systemeinstellungen Netzwerk bietet neben DHCP auch andere Möglichkeiten, die TCP/IP-Parameter festzulegen. Die jeweilige Konfigurationsvariante kann über das Pop-up-Menü *IPv4 konfigurieren* ausgewählt werden. Die Variante *DHCP mit manueller Adresse* erlaubt es, eine IP-Adresse manuell festzulegen und die übrigen TCP/IP-Parameter über DHCP beziehen zu lassen. Die Variante *BOOTP* bezieht die Parameter über das ältere BOOTP-Protokoll, auf dem DHCP aufbaut. Die Variante *manuell* erlaubt schließlich die vollständig manuelle Konfiguration der IP-Parameter.

Abhängig von der Art der gewählten Netzwerkschnittstelle (z. B. Ethernet, AirPort, FireWire u. a.) können im Konfigurationsbereich neben TCP/IP-Parametern auch andere Parameter konfiguriert werden.

So kann für Netzwerkschnittstellen vom Typ Ethernet und AirPort die AppleTalk-Protokollfamilie aktiviert werden. Obwohl die meisten Netzwerkdienste mittlererweile auf TCP/IP und nicht mehr auf AppleTalk basieren, benötigt man AppleTalk oft zum einfacheren Auffinden von Netzwerkresourcen wie beispielsweise von AppleShare-Fileservern und Druckern, sofern diese noch nicht Bonjour unterstützen.

Um sich nach der Aktivierung von AppleTalk einen schnellen Überblick über die im eigenen lokalen Netzwerk mittels AppleTalk angekündigten Dienste zu verschaffen, kann man den Befehl atlookup benutzen:

```
$ atlookup
Found 6 entries in zone *
ff01.04.08      HP LaserJet 2200:SNMP Agent
ff01.04.9e      HP LaserJet 2200:LaserWriter
ff01.04.9d      HP LaserJet 2200:LaserJet 2200
ffc0.72.80      Rafaels PowerBook G4:Darwin
ffa7.a8.80      server:Darwin
ffa7.a8.81      server:AFPServer
```

Die Ausgabe des Befehls zeigt eine tabellarische Übersicht über die im Netzwerk verfügbaren AppleTalk Dienste (so genannte network visible entities, NVEs), mit einer Zeile pro Dienst. In jeder Zeile kann man die Netzwerk-ID, die Knoten-ID, die Socket-ID sowie den Dienstnamen und seinen Typ ablesen.

Ab Mac OS X 10.2 kann AppleTalk in den Netzwerk-Systemeinstellungen nur jeweils für eine einzige Schnittstelle gleichzeitig aktiviert sein. Versucht man AppleTalk für eine weitere Schnittstelle zu aktivieren, wird es für die zuerst verwendete Schnittstelle deaktiviert. Für die meisten Anwendungsfälle ist dieses Verhalten ohnehin völlig ausreichend. Das heißt aber nicht, dass man AppleTalk nicht auf mehr als einer Schnittstelle aktivieren kann: Durch die Manipulation der entsprechenden Konfigurationsdateien ist es sogar möglich, einen Mac-OS-X-Rechner als AppleTalk-Router zu konfigurieren! Mehr dazu im Abschn. 3.2.

Für Schnittstellen vom Typ Ethernet lassen sich ab Mac OS X 10.3 bei Bedarf sogar die Übertragungsgeschwindigkeit, der Duplex-Modus sowie die maximale Paketlänge MTU (maximum trasmission unit) manuell innerhalb der Systemeinstellung Netzwerk definieren (Abb. 3.3).

Damit entfällt die Notwendigkeit, für Konfigurationsaufgaben dieser Art auf den Befehl ifconfig zurückgreifen zu müssen. Dieser sollte nunmehr lediglich dazu benutzt werden, sich einen detaillierten Überblick über die Anzahl der vom System erkannten Netzwerkschnittstellen und ihre aktuelle Konfiguration zu verschaffen:

```
$ ifconfig -a
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 ::1 prefixlen 128
        inet6 fe80::1 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xff000000
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::20a:95ff:feaf:30be prefixlen 64 scopeid 0x4
        inet 192.168.0.13 netmask 0xffffff00 broadcast 192.168.0.255
        ether 00:0a:95:af:30:be
        media: autoselect (1000baseTX <full-duplex>) status: active
        supported media: none autoselect 10baseT/UTP <half-duplex>
                         10baseT/UTP <full-duplex> 10baseT/UTP
                         <full-duplex,hw-loopback> 100baseTX <half-duplex>
                         100baseTX <full-duplex> 100baseTX
                         <full-duplex,hw-loopback> 1000baseTX
                         <full-duplex> 1000baseTX
                         <full-duplex,hw-loopback> 1000baseTX
                         <full-duplex,flow-control> 1000baseTX
                         <full-duplex,flow-control,hw-loopback>
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::20a:95ff:fef4:bde1 prefixlen 64 scopeid 0x5
        inet 192.168.0.14 netmask 0xffffff00 broadcast 192.168.0.255
        ether 00:0a:95:f4:bd:e1
        media: autoselect status: active
        supported media: autoselect
fw0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 4078
```

Abb. 3.3. Ab Mac OS X 10.3 können Ethernet-Schnittstellenparameter wie MTU und Übertragungsgeschwindigkeit in den Netzwerk-Systemeinstellungen konfiguriert werden.

```
        lladdr 00:0a:95:ff:fe:af:30:be
        media: autoselect <full-duplex> status: inactive
        supported media: autoselect <full-duplex>
```

Den physischen Netzwerkschnittstellen Ethernet, AirPort und FireWire entsprechen im obigen Beispiel die logischen Schnittstellenbezeichnungen en0, en1 und fw0. Neben den Konfigurationsparametern für das weit verbreitete Protokoll IP Version 4 (IPv4) findet man in der Ausgabe von ifconfig zu jeder Netzwerkschnittstelle auch jeweils eine Konfigurationszeile für IPv6.

Direkte Änderungen der Schnittstellenkonfiguration mit Hilfe von `ifconfig` sind zwar möglich, aber flüchtig, d. h. sie gehen nach einem Neustart verloren und werden auch nicht in der Systemeinstellung Netzwerk wiedergegeben.

Dieses für Anwender anderer Unix-basierter Betriebssysteme vielleicht etwas überraschende Verhalten hängt mit der Implementierung der Schnittstellenkonfiguration in Mac OS X zusammen. Jedwede Änderungen der Schnittstellenparameter in der Systemeinstellung Netzwerk werden in Mac OS X von einem Daemon – `configd` – überwacht. Dieser Daemon ist dediziert für die Überwachung und Änderung der Netzwerkkonfiguration des Systems verantwortlich. Seine Aufgabe erledigt `configd` mit Hilfe einer Reihe von Konfigurationsagenten, die er als Plugins beim Start lädt[2].

Die Systemeinstellung Netzwerk schreibt Änderungen der Schnittstellenparameter in einer Voreinstellungsdatei[3]. `PreferencesMonitor`, ein Plugin von `configd`, überwacht diese Änderungen und veranlasst die entsprechende Änderung der aktiven Schnittstellenparameter. Um diese Änderung zu bewirken, verwendet `configd` wiederum eines oder mehrere seiner restlichen Plugins, z. B. `ATconfig`, `IPConfiguration` oder `IP6Configuration`.

Besonders interessant ist dabei, dass `configd` mit Hilfe seines `Kernel-EventMonitor`-Plugins den Verbindungsstatus von Netzwerkschnittstellen überwachen kann. Auf diese Weise kann das System auf Änderungen des Verbindungsstatus, also z. B. auf einen Verbindungsabbruch aufgrund eines vom Benutzer abgezogenen Kabels, reagieren und z. B. automatisch die Routing-Tabelle anpassen.

3.2 IP-Routing

Vor dem eigentlich Absenden eines Pakets muss ein Host entscheiden, welchen Weg das Paket nehmen muss, um seinen Empfänger auch wirklich zu erreichen. Das dazu verwendete Entscheidungsverfahren nennt man Routing.

In Mac OS X basiert Routing auf einer im Normalfall automatisch verwalteten Tabelle. Diese so genannte Routing-Tabelle kann man entweder mit dem *Netzwerk-Dienstprogramm* oder mit dem Befehl `netstat` ausgeben:

```
$ netstat -rnf inet
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            192.168.0.1        UGSc        4        5    en0
127                127.0.0.1          UCS         0        0    lo0
127.0.0.1          127.0.0.1          UH          9     2616    lo0
169.254            link#4             UCS         0        0    en0
192.168.0          link#4             UCS         3        0    en0
192.168.0.1        0:3:93:e0:60:c2    UHLW        4        2    en0   1144
192.168.0.4        0:3:93:68:6d:d0    UHLW        1      119    en0    173
192.168.0.11       127.0.0.1          UHS         0        0    lo0
192.168.0.255      ff:ff:ff:ff:ff:ff  UHLWb       0        6    en0
```

[2] /System/Library/SystemConfiguration/
[3] /Library/Preferences/SystemConfiguration/preferences.plist

Die Ausgabe des Netzwerk-Dienstprogramms (Abb. 3.4) und die Ausgabe von `netstat` sind natürlich identisch.

Im Gegensatz zu vielen anderen Systemen ist es unter Mac OS X normalerweise nicht erforderlich, die Routing-Tabelle selbst zu bearbeiten. Konfigurierte Netzwerkschnittstellen werden automatisch in die Routing-Tabelle aufgenommen.

Das obige Beispiel zeigt eine sehr einfache Routing-Tabelle mit nur einer einzigen aktiven Netzwerkschnittstelle (en0, entspricht der integrierten Ethernet-Schnittstelle) und nur einem Router. Die von `netstat` ausgegebene Routing-Tabelle enthält die Spalten Destination, Gateway, Flags, Refs, Use, Netif und Expire. Für das grundsätzliche Verständnis der Routing-Tabelle reicht es dabei, die Spalten Destination, Gateway und Netif (network interface) zu betrachten.

Jede Zeile der Tabelle definiert den Weg zu einem bestimmten Host oder einem bestimmten Netzwerk (Destination):

- Die erste Zeile der obigen Routing-Tabelle definiert die so genannte *Default-Route* (interne Bezeichnung 0.0.0.0). An die hier angegebene IP-Adresse des Routers (192.168.0.1) werden alle IP-Pakete zugestellt, für die es keinen anderen explizit definierten Weg gibt. Der Router kann in unserem Beispiel über die Netzwerkschnittstelle en0 erreicht werden:

```
Destination        Gateway            Netif
default            192.168.0.1          en0
...
```

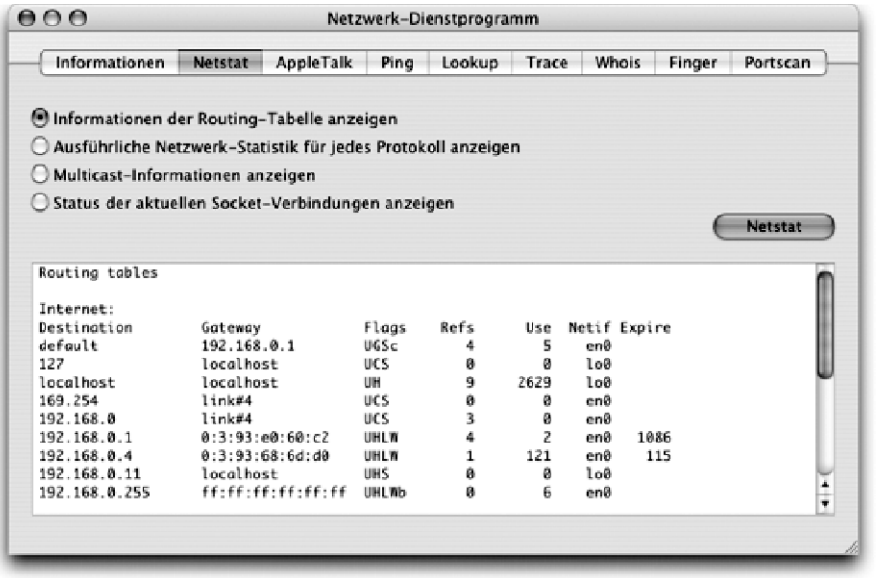


Abb. 3.4. Das Netzwerk-Dienstprogramm ist ein (sehr einfaches) grafisches Frontend für den Befehl netstat.

- Die zweite und dritte Zeile stellen sicher, dass alle an das Netzwerk 127.0.0.0 bzw. an die spezielle IP-Adresse 127.0.0.1 (localhost) adressierten Pakete über die so genannte Loopback-Netzwerkschnittstelle (lo0) ausgeliefert werden. Die Loopback-Schnittstelle ist eine in Software implementierte Pseudoschnittstelle, die es lokalen Prozessen ermöglicht, IP-basierte Kommunikation ohne den Umweg über das Netzwerk abzuwickeln:

```
Destination        Gateway            Netif
...
127                127.0.0.1            lo0
127.0.0.1          127.0.0.1            lo0
...
```

- Die vierte Zeile definiert, dass alle an das Netzwerk 169.254.0.0 adressierten Pakete direkt über die Netzwerkschnittstelle en0 auszuliefern sind, und nicht etwa über einen Router. Das Netzwerk 169.254.0.0 umfasst genau die im Bonjour-Verfahren automatisch zugeteilten IP-Adressen, die nur im lokalen Netzwerk gültig sind:

```
Destination        Gateway            Netif
...
169.254            link#4               en0
...
```

- Die fünfte Zeile definiert, dass alle an das Netzwerk 192.168.0.0 adressierten Pakete ebenfalls direkt über die Netzwerkschnittstelle en0 auszuliefern sind. 192.168.0.0 ist das in unserem Beispiel verwendete private IP-Netzwerk:

```
Destination        Gateway            Netif
...
192.168.0          link#4               en0
...
```

- Die sechste Zeile definiert, dass alle an den Host 192.168.0.1 (den Router) adressierten Pakete direkt über die Netzwerkschnittstelle en0 an das Gerät mit der Ethernetadresse 0:3:93:e0:60:c2 auszuliefern sind:

```
Destination        Gateway            Netif
...
192.168.0.1        0:3:93:e0:60:c2      en0
...
```

- Die siebte Zeile definiert, dass alle an den Host 192.168.0.4 adressierten Pakete direkt über die Netzwerkschnittstelle en0 an das Gerät mit der Ethernetadresse 0:3:93:68:6d:d0 auszuliefern sind:

```
Destination        Gateway            Netif
...
192.168.0.4        0:3:93:68:6d:d0      en0
...
```

- Die achte Zeile definiert schließlich, dass alle an den Host 192.168.0.11 (die eigene IP-Adresse) adressierten Pakete *nicht* über die Netzwerkschnittstelle en0, sondern über die Loopback-Netzwerkschnittstelle lo0 auszuliefern sind. Auf diese Weise werden an sich selbst adressierte Pakete auch

dann nicht über das Netzwerk gesendet, wenn die externe IP-Adresse statt der speziellen localhost-Adresse 127.0.0.1 angegeben wurde:

```
Destination        Gateway            Netif
...
192.168.0.11       127.0.0.1            lo0
...
```

- Die letzte Zeile definiert dann noch, dass lokale Broadcasts, also an alle lokalen Hosts adressierte Pakete als Ethernet-Broadcasts (Ethernetadresse ff:ff:ff:ff:ff:ff) über die Netzwerkschnittstelle en0 auszuliefern sind.

```
Destination        Gateway            Netif
...
192.168.0.255      ff:ff:ff:ff:ff:ff    en0
```

Eventuell vorhandene weitere Netzwerkschnittstellen sind im Auslieferungszustand aktiviert. Sie können mit Hilfe der allgemeinen Konfigurationsansicht in den Systemeinstellungen Netzwerk beliebig deaktiviert und wieder reaktiviert werden, ohne dass es eines Neustarts bedürfte.

Die Routing-Tabelle wird bei Aktivierung von weiteren Netzwerkschnittstellen automatisch um weitere Einträge ergänzt. Ein Rechner mit zwei Netzwerkschnittstellen hat eine gegenüber einem Rechner mit nur einer Netzwerkschnittstelle um mindestens zwei Einträge längere Routing-Tabelle:

```
$ netstat -rnf inet
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            192.168.0.1        UGSc        4        8    en0
127                127.0.0.1          UCS         0        0    lo0
127.0.0.1          127.0.0.1          UH         11     3622    lo0
169.254            link#4             UCS         0        0    en0
192.168.0          link#4             UCS         5        0    en0
192.168.0.1        link#4             UHLW        2        0    en0
192.168.0.4        0:3:93:68:6d:d0    UHLW        1        3    en0   1193
192.168.0.10       127.0.0.1          UHS         0        4    lo0
192.168.1          link#5             UCS         0        0    en1
192.168.1.1        127.0.0.1          UHS         0        8    lo0
```

Die vorletzte Zeile dieses zweiten Beispiels definiert, dass das Netzwerk 192.168.1.0 über die zweite Ethernet-Schnittstelle – Netzwerkschnittstelle en1 – erreicht werden kann. Die letzte Zeile definiert, dass die an den Host selbst über die Schnittstellenadresse 192.168.1.1 adressierten IP-Pakete direkt über die Loopback-Schnittstelle, also ohne Umweg über das Ethernet, zugestellt werden können:

```
Destination        Gateway            Netif
...
192.168.1          link#5               en1
192.168.1.1        127.0.0.1            lo0
```

Obwohl hier zwei Netzwerkschnittstellen aktiviert wurden, bleibt es bei einem Default-Route-Eintrag – genauso wie im ersten Beispiel. Das ist vielleicht etwas überraschend, da in der Systemeinstellung Netzwerk für jede

Netzwerkschnittstelle ein Router angegeben werden musste, über den nicht direkt angeschlossene Hosts erreicht werden können.

Da jedoch protokollbedingt immer nur eine Default-Route existieren kann, muss Mac OS X aus den durch die Schnittstellenkonfiguration bekannten Routern einen designierten Standardrouter auswählen. Das Betriebssystem bedient sich dazu der vom Benutzer vorgegebenen Prioritäten der Netzwerkschnittstellen und verwendet für die Default-Route denjenigen Router, der für die Netzwerkschnittstelle mit der höchsten Priorität konfiguriert wurde. Netzwerkschnittstellen ohne Verbindung zu einem Netzwerk bleiben dabei natürlich unberücksichtigt.

Die Prioritäten der Netzwerkschnittstellen legt der Benutzer einfach mit Hilfe der allgemeinen Konfigurationsansicht in Systemeinstellungen Netzwerk fest (Abb. 3.5). In dieser Ansicht lassen sich Netzwerkschnittstellen nicht nur deaktivieren und reaktivieren, sondern durch das Verschieben mit der Maus nach Prioritäten ordnen.

In der im obigen Beispiel dargestellten Routing-Tabelle wurde für die Default-Route der für die integrierte Ethernet-Schnittstelle (Netzwerkschnittstelle en0) konfigurierte Router 192.168.0.1 verwendet, weil dieser eine höhere Priorität als der Ethernet-Schnittstelle auf der PCI-Karte (Netzwerkschnittstelle en1) zugeordnet wurde.

Die automatische Aktualisierung ermöglicht einen sehr einfachen Wechsel zwischen Netzwerkverbindungen, insbesondere zwischen drahtgebundenen und drahtlosen Netzwerken.

Das Betriebssystem legt die Default-Route wie bereits angeschnitten nicht ausschließlich anhand der für die Netzwerkschnittstellen definierten Prioritäten, sondern berücksichtigt auch den Verbindungsstatus der jeweiligen Netzwerkschnittstelle.

Dazu überwacht Mac OS X permanent alle Netzwerkschnittstellen und versucht zu erkennen, ob die Netzwerkschnittstellen auch tatsächlich mit einem Netzwerk verbunden sind. Ist das nicht mehr der Fall (z. B. weil das Ethernet-Kabel abgezogen wurde oder weil der Benutzer sich außer Reichweite einer AirPort-Basisstation begeben hat) und wurde die betroffene Netzwerkschnittstelle bisher für die Default-Route verwendet, dann wird automatisch die bisherige Default-Route deaktiviert und eine neue installiert, welche dann die Netzwerkschnittstelle mit der nächst niedrigeren Priorität verwendet.

Was theoretisch recht kompliziert klingt, ist in der Anwendung sehr einfach. Beispielsweise kann man in einem sowohl mit einem drahtlosen AirPort- als auch mit einem kabelgebundenen Ethernet-Netzwerk ausgestatteten Büro abwechselnd beide Netzwerke nutzen, ohne manuell zwischen zwei Konfigurationen umschalten (oder gar neu starten!) zu müssen.

Man ordnet dazu die Ethernet-Schnittstelle einfach oberhalb der AirPort-Schnittstelle an und weist ihr damit eine höhere Priorität zu. Steckt man das Ethernet-Kabel dann ein, wird (unter der Voraussetzung, dass die Ethernet-

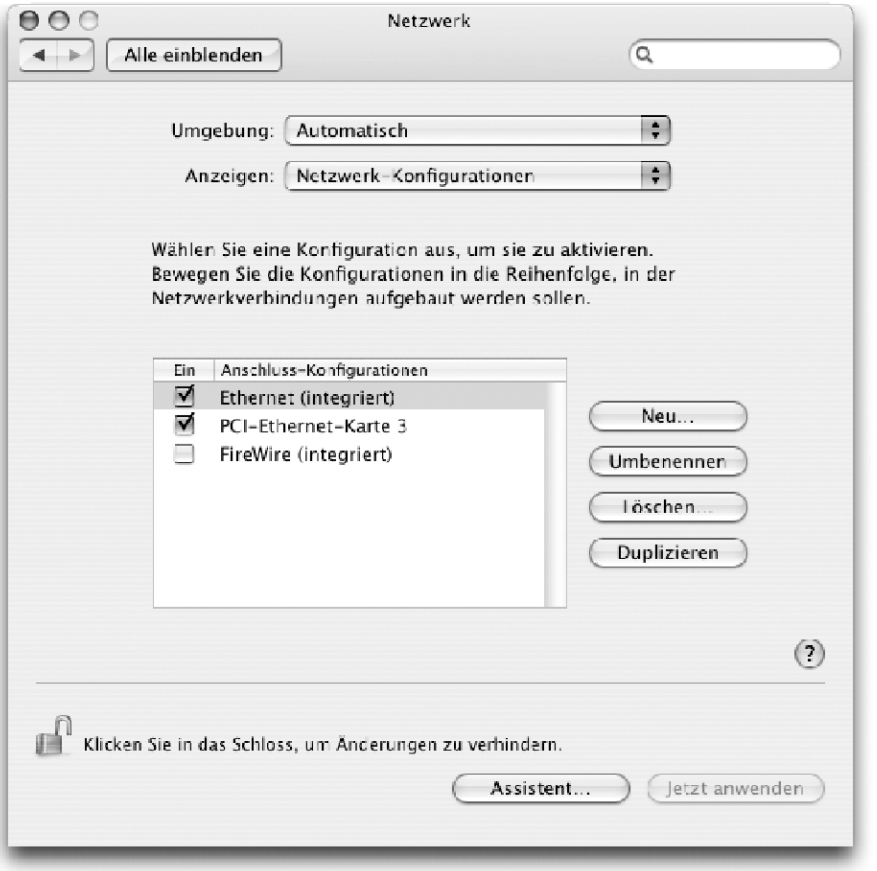


Abb. 3.5. In der allgemeinen Konfigurationsansicht lassen sich Netzwerkschnittstellen aktivieren und deaktivieren sowie nach Priorität ordnen.

Schnittstelle korrekt konfiguriert ist) die Ethernet-Schnittstelle sowohl für Verbindungen zu lokalen Hosts als auch für Verbindungen zu externen Hosts (über den Router im Ethernet-Netzwerk) verwendet.

Ist man gleichzeitig in Reichweite einer AirPort-Basisstation, dann wird die AirPort-Schnittstelle ausschließlich für Verbindungen zu lokalen Hosts des so angeschlossenen drahtlosen Netzwerks verwendet, die nicht gleichzeitig über das Ethernet-Netzwerk erreichbar sind. Das ändert sich jedoch automatisch, wenn man das Ethernet-Kabel absteckt: In diesem Fall wird nach einer kurzen Verzögerung die Default-Route geändert, so dass alle Verbindungen, inklusive der Verbindungen zu externen Hosts, über den Router des drahtlosen AirPort-Netzwerks hergestellt werden.

Mit Hilfe von `netstat` kann man das leicht überprüfen (Ausgabe gekürzt):

```
$ netstat -rnf inet
Routing tables
```

```
Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            192.168.0.1        UGSc        2        3    en0
...

... (Ethernet-Kabel abstecken und kurz warten) ...

$ netstat -rnf inet
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif Expire
default            192.168.0.1        UGSc        1        4    en1
...
```

Man sieht, dass nach dem Abstecken des Ethernet-Kabels die Default-Route automatisch im Hinblick auf die zu benutzende Netzwerkschnittstelle (en1 statt en0) modifiziert wurde. Die Adresse des Routers (Gateway) hat sich in diesem Beispiel nicht geändert, weil der Router in diesem speziellen Fall über beide Netzwerke erreichbar ist.

Wie man sieht, wird die Routing-Tabelle im Normalfall vollautomatisch vom System verwaltet, ohne dass eine manuelle Konfiguration erforderlich wäre. Nur in ganz bestimmten Ausnahmefällen muss man manuell Einträge hinzuzufügen oder entfernen. Das ist regelmäßig z. B. dann der Fall, wenn der Rechner als Router bzw. als Gateway eingerichtet werden soll, der zwei oder mehr Netzwerke miteinander verbindet.

Nehmen wir dazu an, dass wir über zwei lokale Ethernet-Netzwerke verfügen: ein Netzwerk A mit der Netzwerk-Adresse 192.168.0.0 und ein Netzwerk B mit der Netzwerkadresse 192.168.1.0.

Um nun eine netzwerkübergreifende Kommunikation zu ermöglichen, müssen wir sie mit Hilfe eines Routers bzw. eines Gateways verbinden. Diese Funktion kann ein Host übernehmen, der über mindestens zwei Netzwerkschnittstellen verfügt und damit mit beiden Netzwerken gleichzeitig verbunden ist.

Falls wir einen Mac-OS-X-Rechner mit beiden Netzwerken verbinden und die beiden Schnittstellen über die Systemeinstellung Netzwerk konfigurieren, dann kann dieser eine Rechner automatisch und ohne dass weitere Konfigurationsschritte notwendig wären mit Hosts in beiden Netzwerken kommunizieren.

Um zu erreichen, dass beliebige Hosts aus dem Netzwerk A mit beliebigen Hosts im Netzwerk B kommunizieren können, müsste dieser Mac-OS-X-Rechner die Funktion eines Routers bzw. eines Gateways übernehmen.

Obwohl Mac OS X die Weiterleitung von Paketen von Hause aus unterstützt, ist diese Funktion im Auslieferungszustand nicht aktiviert. Um sie zu aktivieren, genügt es, den entsprechenden Parameter im Mac-OS-X-Betriebssystemkern zu setzen:

```
$ sudo sysctl -w net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
```

Um diesen Befehl nicht nach jedem Neustart manuell ausführen zu müssen, kann man ihn in ein StartupItem einbetten. Unter Mac OS X 10.3 war der nötige Code bereits im StartupItem „Network“ integriert und musste nur noch durch das Setzen der IPFORWARDING in /etc/hostconfig aktiviert werden. In Mac OS X 10.4 wurde das StartupItem „Network“ entfernt, der entsprechende Code lässt sich aber natürlich in ein eigenes StartupItem integrieren.[4] Das entsprechende StartupItem legt man am besten in /Library/StartupItems an:

```
...
if [ "${IPFORWARDING:=-NO-}" = "-YES-" ]
then
        sysctl -w net.inet.ip.forwarding=1 > /dev/null
else
        sysctl -w net.inet.ip.forwarding=0 > /dev/null
fi
...
```

Will man die Aktivierung der Paketvermittlung ähnlich wie in Mac OS X 10.3 mit Hilfe der Variablen IPFORWARDING steuern, muss diese Variable in der Datei `/etc/hostconfig` definiert werden. Hat man ein StartupItem mit dem obigen Code erstellt, schaltet die Definition `IPFORWARDING=-NO-` die Paketvermittlung ab und versetzt den Rechner damit in den Auslieferungszustand. Die Definition `IPFORWARDING=-YES-` schaltet dagegen die Paketvermittlung ein.

Nach dem Einschalten der Paketvermittlung auf unserem Verbindungsrechner müssen wir den Rechnern in den beiden verbundenen Netzwerken nur noch mitteilen, dass Pakete an Hosts in dem jeweils anderen Netzwerk über den Verbindungsrechner zuzustellen sind.

Im einfachsten Fall tragen wir dazu die IP-Adresse des Verbindungsrechners als Router- bzw. Gateway-Adresse ein. Danach werden alle Pakete, die nicht an Hosts im lokalen Netzwerk adressiert werden, an unseren Mac-OS-X-Router geschickt. Dieser leitet sie dann einfach an das jeweils andere Netzwerk weiter.

Falls die IP-Adresse der Ethernet-Schnittstelle unseres Routers im Netzwerk A 192.168.0.1 und die IP-Adresse seiner Ethernet-Schnittstelle im Netzwerk B 192.168.1.1 lautet, ist bei allen Hosts im Netzwerk A die IP-Adresse 192.168.0.1 und bei allen Hosts im Netzwerk B die IP-Adresse 192.168.1.1 als Router-Adresse einzutragen.

Komplizierter wird es, wenn nicht zwei, sondern drei oder mehr Netzwerke miteinander verbunden werden; wenn also beispielsweise das Netzwerk B über einen zweiten Router mit einem Netzwerk C verbunden wird, über das auch noch weitere Netzwerke erreichbar sind. In solchen Fällen ist es machmal notwendig, einen statischen Eintrag in der Routing-Tabelle zu erzeugen.

Um festzulegen, dass alle Hosts im hypothetischen Netzwerk C mit der Adresse 192.168.2.0 ausschließlich über den Router mit der IP-Adresse 192.168.1.254 zu erreichen sind, kann man den Befehl `route` verwenden:

[4] siehe Anhang A.2

```
$ sudo route -n add -net 192.168.2 192.168.1.254
add net 192.168.1: gateway 192.168.0.1
$ netstat -rnf inet
...
192.168.2     192.168.1.254     UGSc      0      0    en1
```

Falls der Eintrag nicht mehr benötigt wird, kann er mit diesem Befehl auch gelöscht werden:

```
$ sudo route -n delete -net 192.168.2
delete net 192.168.2
```

Zu beachten ist, dass ein auf diese Weise erzeugter Eintrag flüchtig ist, d. h. nach dem nächsten Neustart neu angelegt werden muss. Normalerweise wird man hierfür ein eigenes StartupItem anlegen und in `/Library/StartupItems/` ablegen. Dabei sollte man die Aktivierung des Eintrags vom Zustand der Variablen `IPFORWARDING` abhängig machen oder `/etc/hostconfig` um eine weitere Variable ergänzen. Der Skriptcode könnte dann etwa so lauten:

```
...
if [ "${IPFORWARDING:=-NO-}" = "-YES-" ]
then
        route -n add -net 192.168.1 192.168.0.1 > /dev/null
fi
...
```

3.3 AppleTalk-Routing

Ab Mac OS X 10.2 kann man das AppleTalk-Protokoll in der Systemeinstellung Netzwerk nur noch für eine einzige Netzwerkschnittstelle aktivieren. Um die AppleTalk-Unterstützung für mehr als eine Netzwerkschnittstelle zu aktivieren, muss man die Voreinstellungsdatei der Systemeinstellung Netzwerk manuell verändern.

In dieser Voreinstellungsdatei lässt sich ein Mac-OS-X-Rechner dann auch als AppleTalk-Router konfigurieren. Ein solcher AppleTalk-Router kann den angeschlossenen AppleTalk-Hosts DDP-Adressen aus festgelegten Adressenbereichen zuweisen sowie eine oder mehrere AppleTalk-Zonen definieren. AppleTalk-Zonen können helfen, große AppleTalk-Netzwerke für die Anwender übersichtlicher zu gestalten.

Um alle diese Funktionen zu nutzen, muss man allerdings die von der Systemeinstellung Netzwerk manipulierte Voreinstellungsdatei `preferences.plist`[5] direkt verändern und anschließend einen Neustart durchführen. Da fehlerhafte Änderungen gravierende Folgen haben können, sollte man vor der manuellen Veränderung dieser Datei immer zuerst eine Sicherheitskopie erstellen, auf die man im Notfall zurückgreifen kann.

Um AppleTalk-Unterstützung bzw. AppleTalk-Routing für eine Netzwerkschnittstelle zu aktivieren, muss man die Konfigurationsparameter dieser

[5] siehe /Library/Preferences/SystemConfiguration/

Schnittstelle um einige Attribute ergänzen. Da es sich bei der Voreinstellungsdatei um eine Datei im Property-List-Format handelt, kann für die Bearbeitung der Datei entweder ein beliebiger Texteditor, oder der zum Lieferumfang der von Apple bereitgestellten „Developer Tools“ gehörende „Property List Editor“ verwendet werden (Abb. 3.6).

In der Voreinstellungsdatei sind die Konfigurationsparameter aller Netzwerkschnittstellen gespeichert. Hat man in der Systemeinstellung Netzwerk mehrere „Umgebungen“ definiert, gibt es in der Voreinstellungsdatei für jede Netzwerkschnittstelle mehrere Einträge, die den Konfigurationen der Schnittstelle in der jeweiligen Umgebung entsprechen. Bevor man anfängt, die Konfigurationsparameter zu editieren, muss man daher u. U. erst die Identifikatoren der zu konfigurierenden Schnittstellen in der aktuellen Umgebung finden.

Die Konfigurationsparameter der Netzwerkschnittstellen sind im Dictionary `NetworkServices` abgelegt. Wurden außer der Standardumgebung „Automatic“ (deutsch „Automatisch“) keine weiteren Umgebungen definiert, enthält dieses Dictionary für jede Netzwerkschnittstelle genau einen Eintrag. Ein solcher Eintrag enthält u. a. das Attribut `UserDefinedName`, mit dessen

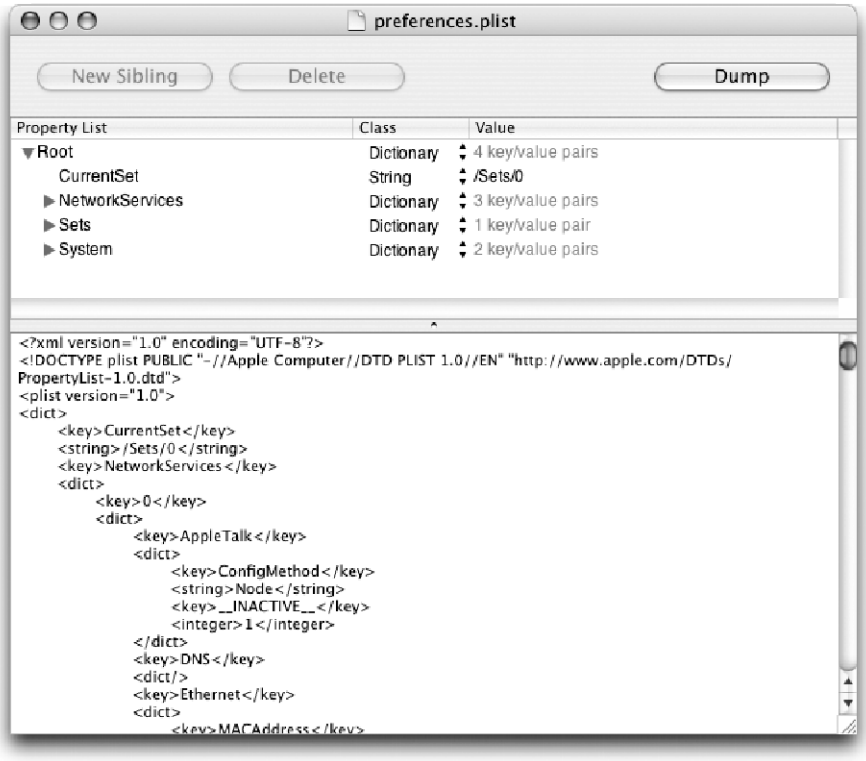


Abb. 3.6. Die Voreinstellungsdatei der Systemeinstellung Netzwerk lässt sich im Property List Editor öffnen und editieren.

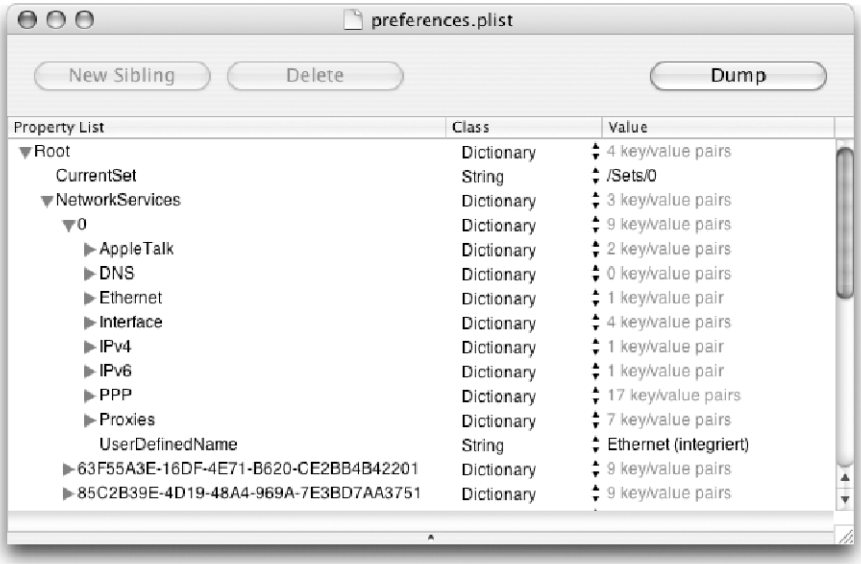


Abb. 3.7. Das Attribut UserDefinedName hilft, den Konfigurationseintrag einer bestimmten Netzwerkschnittstelle zu finden.

Hilfe sich der Eintrag auf einfache Weise eindeutig einer bestimmten Netzwerkschnittstelle zuordnen lässt.

Wurden mehrere Umgebungen definiert, findet man u. U. mehrere Einträge mit dem gleichen `UserDefinedName`. Um den aktiven Eintrag zu finden, muss man die Definition der aktiven Umgebung betrachten. Den Identifikator der aktiven Umgebung kann man dem Attribut `CurrentSet` entnehmen. Die Standardumgebung „Automatisch“ hat dabei normalerweise den Identifikator „0“.

Die Umgebungen selbst sind im Dictionary `Sets` definiert. Jeder Eintrag in diesem Dictionary entspricht einer definierten Umgebung. Hat man die Definition der aktiven Umgebung identifiziert, kann man den entsprechenden Eintrag innerhalb des Dictionaries im *Property List Editor* aufklappen und die Definition betrachten.

Innerhalb der Definition der aktiven Umgebung findet man dann das Dictionary `Services`, das Verweise auf die verwendeten Schnittstellenkonfigurationen in `NetworkServices` enthält. Ein Verweis, der auf die Schnittstellenkonfiguration mit der Bezeichnung `0` zeigt, hat die Form (siehe auch Abb. 3.8):

```
$ cd /Library/Preferences/SystemConfiguration
$ cat preferences.plist
...
      <key>Service</key>
      <dict>
        <key>0</key>
```

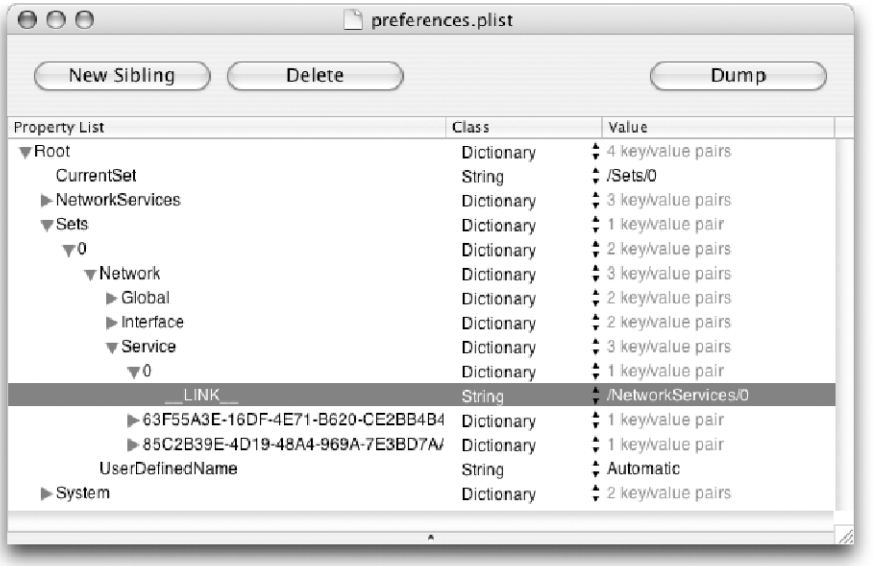


Abb. 3.8. Die Definition der Umgebung „Automatic“ enthält einen Verweis auf die Schnittstellendefinition mit der Bezeichnung 0.

```
        <dict>
          <key>__LINK__</key>
          <string>/NetworkServices/0</string>
        </dict>
      </dict>
...
```

Mit Hilfe dieser Verweise lässt sich leicht herausfinden, welche Schnittstellenkonfiguration bearbeitet werden muss.

Hat man die gewünschte Schnittstellenkonfiguration gefunden, muss man den Schnittstelleneintrag im Bereich AppleTalk bearbeiten. Normalerweise ist dort festgelegt, dass der Host als normaler AppleTalk-Knoten, d. h. nicht als Router, arbeiten soll:

```
...
<key>NetworkServices</key>
<dict>
<key>0</key>
<dict>
<key>AppleTalk</key>
<dict>
<key>ConfigMethod</key>
<string>Node</string>
</dict>
...
```

Ist für die Netzwerkschnittstelle AppleTalk nicht aktiv, enthält das AppleTalk-Dictionary zusätzlich noch einen Eintrag `__INACTIVE__` vom Typ Integer mit Wert 1 (Abb. 3.9):

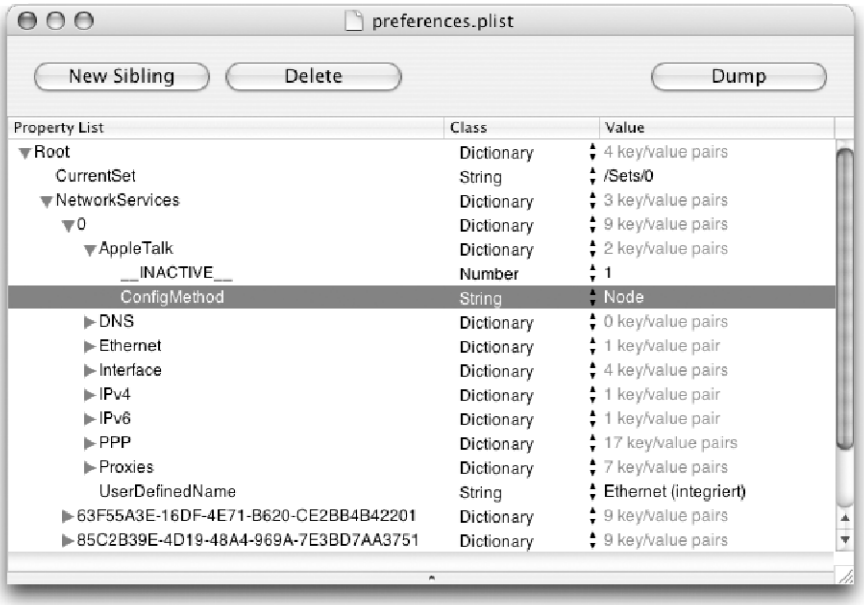


Abb. 3.9. Die Netzwerkschnittstelle wurde als inaktiver AppleTalk-Knoten konfiguriert.

```
...
<key>NetworkServices</key>
<dict>
<key>0</key>
<dict>
<key>AppleTalk</key>
<dict>
<key>ConfigMethod</key>
<string>Node</string>
<key>__INACTIVE__</key>
<integer>1</integer>
</dict>
...
```

Die Systemeinstellungen Netzwerk lassen nicht zu, dass man auf mehr als einer Netzwerkschnittstelle das AppleTalk-Protokoll aktiviert. Manuell kann man das aber leicht machen, indem man die beiden Zeilen, die dem Eintrag `__INACTIVE__` entsprechen, einfach aus der Voreinstellungsdatei löscht und anschließend einen Neustart ausführt. Im *Property List Editor* genügt es, die entsprechende Zeile zu selektieren und den Knopf „Delete“ zu drücken.

Zusätzlich kann man die AppleTalk-Routing-Funktionen aktivieren, indem man den Knotentyp ändert. Um die Seed-Router-Funktion zu aktivieren, muss man die `ConfigMethod` von `Node` in `SeedRouter` ändern. Gibt es schon einen anderen AppleTalk-Seed-Router im mit der zu konfigurierenden Netzwerkschnittstelle verbundenen Netzwerk, kann man alternativ ein einfa-

ches AppleTalk-Routing ohne Seed-Router-Funktionalität aktivieren, indem man den Wert `Router` angibt.

Ein AppleTalk-Seed-Router sorgt dafür, dass alle AppleTalk-Hosts im angeschlossenen Netzwerk DDP-Adressen aus einem festgelegten Bereich verwenden. Gleichzeitig kann er AppleTalk-Zonen definieren, denen die AppleTalk-Hosts übersichtshalber zugeordnet werden können.

Die Konfiguration eines AppleTalk-Seed-Routers muss man daher um die zwei Parameter `SeedNetworkRange` (Typ Array mit Elementen vom Typ Number) und `SeedZones` (Typ Array mit Elementen vom Typ String) ergänzen. Mit `SeedNetworkRange` lässt sich der Bereich der DDP-Netzwerkadressen festlegen, der von den an das jeweilige Netzwerk angeschlossenen AppleTalk-Hosts verwendet werden soll. Mit `SeedZones` wird die Liste der AppleTalk-Zonen definiert, die von den angeschlossenen Hosts verwendet werden können. Die im `SeedZones`-Array als erstes Element (Index 0) angegebene Zone wird als Standardzone aufgefasst, in der alle Hosts auftauchen, denen manuell keine andere Zone zugewiesen worden ist.

Eine einfache Seed-Router-Konfiguration für das Netzwerk mit der Nummer 1 und zwei Zonen kann z. B. wie folgt aussehen (siehe auch Abb. 3.10):

```
...
<key>NetworkServices</key>
<dict>
<key>0</key>
<dict>

<key>Apple\-Talk</key>
<dict>
<key>ConfigMethod</key>
<string>SeedRouter</string>
<key>SeedNetworkRange</key>
<array>
<integer>1</integer>
<integer>1</integer>
</array>
<key>SeedZones</key>
<array>
<string>Privat</string>
<string>Öffentlich</string>
</array>
</dict>
...
```

Die Voreinstellungsdatei ist nur für root schreibbar. Um sie direkt zu editieren, muss man sich also entweder als root einloggen (explizite Freischaltung der root-Kennung mit Hilfe des *NetInfo-Managers* erfoderlich) oder einen Editor verwenden, der eine zusätzliche Authentifizierung bei ungenügenden Zugriffsrechten unterstützt. Am einfachsten geht es jedoch, wenn man einer Administrator-Kennung mit Hilfe des Finders den Schreibzugriff auf die Datei erlaubt und sie dann einfach im *Property List Editor* öffnet.

Nachdem man die Datei modifiziert hat, muss man einen Neustart ausführen. Mit Hilfe des Befehls `atlookup` kann man verifizieren, ob die Apple-

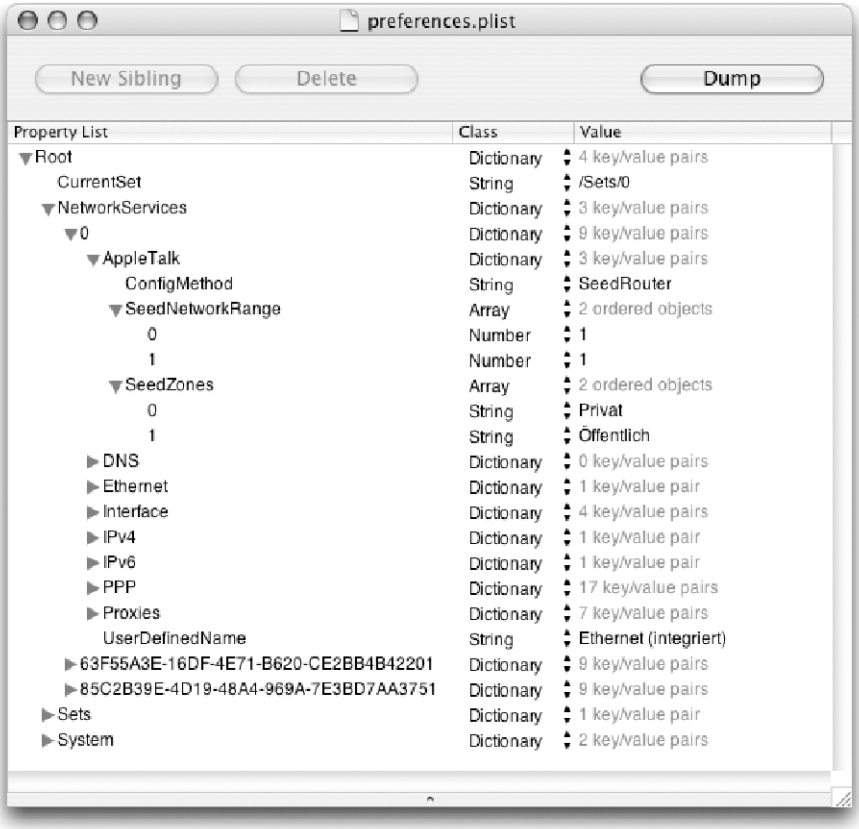


Abb. 3.10. Eine einfache Seed-Router-Konfiguration mit zwei Zonen.

Talk-Hosts die erwarteten DDP-Adressen erhalten und ob sie in der Standardzone auftauchen:

```
$ atlookup
Found 7 entries in zone Privat
0001.04.08      HP LaserJet 2200:SNMP Agent
0001.04.9e      HP LaserJet 2200:LaserWriter
0001.04.9d      HP LaserJet 2200:LaserJet 2200
0001.04.9a      HP LaserJet 2200:HP Zoner Responder
0001.8f.80      Tiger Client:Darwin
0001.d2.80      Tiger Server:Darwin
0001.76.80      iMac G4:Darwin
```

Um die AppleTalk-Routing-Tabelle auszugeben, kann man den Befehl `appletalk` verwenden:

```
$ appletalk -j

-------- Appletalk Configuration -----------
                              Network:
 I/F  State        Range       Node      Default Zone
```

```
 ---- ------------ ----------- --------- ------------------
*en0  Online           1-1         1:210 Privat
```

Die vom Seed-Router definierten Zonen kann man dazu nutzen, AppleTalk-basierte Netzwerkdienste wie beispielsweise Drucker oder Fileserver zu gruppieren. Man weist dazu jedem AppleTalk-Host die Zone zu, in der er auftauchen soll. AppleTalk-Hosts, denen keine Zone explizit zugewiesen wurde, tauchen in der Standardzone (Default Zone) auf.

Die Zuordnung der Hosts zu AppleTalk-Zonen erfolgt dabei dezentral auf der Clientseite. Im klassischen Mac OS kann man die Zugehörigkeit zu einer AppleTalk-Zone im AppleTalk-Kontrollfeld festlegen. Unter Mac OS X trifft man die Festlegung entsprechend in der Systemeinstellung Netzwerk (Abb. 3.11).

Abb. 3.11. Die Zugehörigkeit zu einer AppleTalk-Zone kann man in der Systemeinstellung Netzwerk vornehmen.

Startet man AppleTalk in der Kommandozeile, kann man explizit die AppleTalk-Zone angeben, zu der man gehören will (mit `appletalk -d` wird AppleTalk angehalten, falls es bereits aktiv war):

```
$ sudo appletalk -d
Password:
Tiger-Client:~ admin$ sudo appletalk -u en0
Current zone for interface en0 is Privat
Zones found:
 1: Privat
 2: ?ffentlich

Enter <return> to exit.

warning: this program uses gets(), which is unsafe.
Zone Number? 2
Default zone changed to ?ffentlich
```

Wurde AppleTalk sowohl in der Systemeinstellung Netzwerk als auch im Verzeichnisdienste-Dienstprogramm aktiviert, erscheinen AppleTalk-Zonen als Verzeichnisse im Netzwerk-Browser des Mac OS X Finders (Abb. 3.12). Innerhalb dieser Verzeichnisse werden dann diesen Zonen zugewiesene Fileserver angezeigt.

AppleTalk-fähige Drucker kann man mit Hilfe des Drucker-Dienstprogramms verwenden. Auch dieses Programm unterstützt AppleTalk-Zonen und erlaubt das zonenweise Suchen nach aktiven Druckern (Abb. 3.13).

Abb. 3.12. Wurde AppleTalk sowohl in der Systemeinstellung Netzwerk als auch im Verzeichnisdienste-Dienstprogramm aktiviert, erscheinen AppleTalk-Zonen als Verzeichnisse im Netzwerk-Browser des Mac OS X Finders.

Abb. 3.13. Das Drucker-Dienstprogramm erlaubt das Durchsuchen von AppleTalk-Zonen nach aktiven Druckern.

3.4 Grundlegende Netzwerkdienste bereitstellen

3.4.1 DHCP

Das Dynamic Host Configuration Protocol (kurz DHCP) wird heutzutage in sehr vielen TCP/IP-basierten Netzwerken eingesetzt. Egal ob Macintosh-Rechner, Windows-basierte PCs oder Drucker – heutzutage können fast alle netzwerkfähigen Geräte als DHCP-Clients konfiguriert werden und auf diese Weise die TCP/IP-Konfigurationsparameter automatisch von einem DHCP-Server beziehen.

In kleineren Unternehmen und in Privathaushalten kommen vielfach kompakte Router zum Einsatz, die einen DHCP-Server beinhalten und automatisch zwischen privaten und öffentlichen IP-Adressen übersetzen (d. h. die Funktion eines NAT-Routers übernehmen können). Diese Geräte dienen als Schnittstelle zum öffentlichen Internet und verfügen zu diesem Zweck über

eine integrierte ISDN- oder eine Ethernet-Schnittstelle zum Herstellen einer WAN-Verbindung über das öffentliche ISDN-Telefonnetz oder DSL (z. T. mittelbar mit Hilfe eines PPPoE[6]-kompatiblen externen DSL-Modems oder mit Hilfe einer bereits integrierten Hardware).

Die meisten dieser Geräte können sehr einfach über eine Web-Schnittstelle mit Hilfe eines Web-Browsers konfiguriert werden. Abbildung 3.14 zeigt die Web-Schnittstelle eines solchen kompakten Routers (Modell Vigor 2000 des taiwanesischen Herstellers DrayTek) zur Konfiguration des eingebauten DHCP-Servers.

Die Konfigurationsmöglichkeiten solcher Router sind im Hinblick auf den DHCP-Server, wie man an diesem Beispiel sieht, begrenzt. Man kann in der Regel nur einen einzigen Pool von Adressen angeben (im Beispiel besteht der Pool aus 244 Adressen aus dem Bereich 192.168.0.10 bis 192.168.0.254). Man kann nur die wichtigsten TCP/IP-Konfigurationsparameter übertragen, nicht aber fortgeschrittene Parameter wie z. B. die IP-Adresse des zu verwendenden Verzeichnisservers. Und schließlich kann man in den meisten Fällen lediglich dynamisches DHCP verwenden, hat also keinen Einfluss darauf, welchem Host welche IP-Adresse zugewiesen wird.

Ähnlichen Beschränkungen unterliegt auch der DHCP-Server der AirPort (Extreme) Basisstationen (Abb. 3.15). Immerhin kann man hier die Lease-Dauer einstellen – allerdings nur, wenn man auf die NAT-Funktion verzichtet.

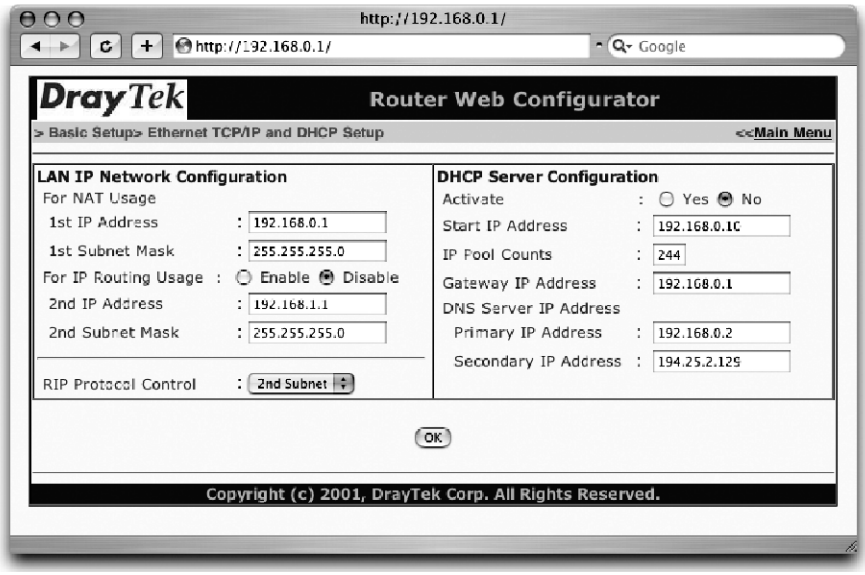


Abb. 3.14. Beispiel der Web-Schnittstelle eines kompakten Routers – DrayTek Vigor 2000 – zur Konfiguration des eingebauten DHCP-Servers.

[6] PPP over Ethernet

Abb. 3.15. Auch der DHCP-Server der AirPort-Extreme-Basisstation bietet zwar für die meisten Fälle ausreichende, aber doch recht beschränkte Einstellungsmöglichkeiten.

Der Funktionsumfang des DHCP-Servers eines solchen Routers bzw. einer AirPort-Basisstation ist natürlich auf den üblichen Einsatzzweck abgestimmt. Er deckt alle Fälle ab, in denen es um die einfache Bereitstellung einer Netzwerkinfrastruktur für ein kleines Netzwerk und dessen Anbindung an das öffentliche Netz geht. Durch die Beschränkung auf das Wesentliche ist der Konfigurationsaufwand minimal und für den Anwender auch ohne Spezialwissen einfach zu bewältigen.

Reicht der Funktionsumfang eines solchen DHCP-Servers nicht aus, muss man einen etwas höheren Konfigurationsaufwand in Kauf nehmen und einen eigenen DHCP-Server aufsetzen. Unter Mac OS X kann man den eingebauten DHCP-Server *bootpd* verwenden.

Am einfachsten lässt sich der eingebaute DHCP-Server über die Systemeinstellung Sharing aktivieren: die Schaltfläche *Internet* führt dort zu einem Dialog, der die Aktivierung des DHCP-Servers erlaubt.

Der DHCP-Server kann auf diese Weise allerdings nur zusammen mit NAT und einem Proxy-DNS-Server aktiviert werden (dazu später noch mehr). Die Einstellungsmöglichkeiten sind sehr rudimentär und auf den bereits weiter oben beschriebenen, sehr verbreiteten Anwendungsfall zugeschnitten: die Verbindung eines kleinen privaten Netzwerks mit der Außenwelt.

Abb. 3.16. In der Systemeinstellung Sharing kann man den DHCP-Server zusammen mit einer NAT-Routing-Funktion und einem Proxy-DNS-Server aktivieren, um ein angeschlossenes Netzwerk auf eine einfache Weise ohne jeglichen Konfigurationsaufwand mit einer einfachen Netzwerkinfrastruktur und einer Verbindung zum Internet zu versorgen.

Mit Hilfe der Systemeinstellung Sharing kann ein Mac OS X demnach auf sehr einfache Weise die Funktion eines NAT-Routers übernehmen. Man muss lediglich die öffentliche (Pop-up-Menü „Verbindung gemeinsam nutzen") und die private (eine oder mehrere Einträge in der Checkbox-Liste „Mit Computern, die Folgendes verwenden") Netzwerkschnittstelle wählen und Startknopf drücken, um den DHCP-Server, NAT und Proxy-DNS automatisch konfigurieren zu lassen und zu starten.

Unmittelbar nach dem Start wird der DHCP-Server aktiv und beginnt über die als privat deklarierten Netzwerkschnittstellen IP-Adressen, die Router-Adresse und die Adresse des DNS-Servers zu verteilen. Der dabei verwendete IP-Adressenpool und die IP-Adresse des eigenen Rechners sind über

die grafische Benutzeroberfläche nicht einstellbar. Als Router-Adresse und DNS-Server-Adresse wird jeweils die Adresse der eigenen privaten Netzwerkschnittstelle benutzt.

Natürlich lässt sich der DHCP-Server auch manuell starten und nach Belieben konfigurieren. Man muss dazu den DHCP-Server per Hand konfigurieren und sollte dann auf die Benutzung von Internet Sharing verzichten, um Konfigurationskonflikte zu vermeiden. Die DHCP-Server-Funktionalität wird unter Mac OS X über den Befehl *booptd* realisiert. Bei diesem Programm handelt es sich um einen typischen Unix Daemon, also um einen Befehl, der typischerweise im Hintergrund, ohne dem Benutzer direkte Interaktionsmöglichkeiten zu bieten, abläuft.

Der Leistungsumfang und die Konfigurationsmöglichkeiten von *bootpd* sind sehr detailliert in der dazugehörigen man-Page beschrieben. Neben DHCP beherrscht er auch BOOTP sowie NetBoot 1.0 und 2.0.

Die Konfiguration des DHCP-Servers erfolgt über die lokale NetInfo-Datenbank mit Hilfe von Einträgen in den NetInfo-Verzeichnissen /config/dhcp und /config/dhcp/subnets.[7] Einige wenige Konfigurationsparameter lassen sich auch direkt über Kommandozeilenparameter steuern. Den Daemon selbst betreffende Optionen wie die detailliertere Protokollierung aller wichtigen Ereignisse (Option *-v*) oder der Vordergrundbetrieb zu Testzwecken (Option *-d*) werden ebenfalls über Kommandozeilenparameter gesteuert.

Das Protokoll wird in die Datei `/var/log/system.log` geschrieben und beinhaltet Einträge zu allen Anfragen von DHCP-Clients und allen Antworten des DHCP-Servers. Die Protokolldatei kann man mit einem beliebigen Texteditor oder aber mit dem Dienstprogramm Konsole betrachten. Alternativ kann man den Befehl

```
$ tail -f /var/log/system.log
...
```

verwenden, um das Ende der Protokolldatei im Auge zu behalten.

Um `bootpd` zu starten, muss der volle (absolute) Pfad des Befehls angegeben werden, da sich dieser Befehl in keinem der Standardbefehlsverzeichnisse befindet und ansonsten nicht gefunden wird.

```
$ echo $PATH
/bin:/sbin:/usr/bin:/usr/sbin
$ bootpd
-bash: bootpd: command not found
```

Root-Rechte (sudo) sind erforderlich, da der Server den privilegierten Port 67 (bootps) benutzen muss, um BOOTP- und DHCP-Anfragen beantworten zu können.

```
$ sudo /usr/libexec/bootpd -d
bootpd[673]: interface en0: ip 192.168.0.10 mask 255.255.255.0
bootpd[673]: interface en1: ip 192.168.1.1 mask 255.255.255.0
default: 192.168.0.1
127 ==> link 0
169.254 ==> link 4
```

[7] siehe auch Abschn. 4.1

```
192.168.0 ==> link 4
192.168.2 ==> link 5
...
```

Wird der DHCP-Server wie hier ohne weitere Optionen gestartet, aktiviert er lediglich BOOTP und kein DHCP. `bootpd` verwendet dazu alle aktiven und konfigurierten Netzwerkschnittstellen des Systems. Zusätzliche Netzwerkschnittstellen muss man immer zuerst konfigurieren, bevor man sie zusammen mit dem DHCP-Server verwenden kann (Abb. 3.17).

Sofern das /machines-Verzeichnis der lokalen NetInfo-Datenbank keine zusätzlichen Einträge zur Beantwortung von BOOTP-Anfragen enthält, beantwortet `bootpd` in diesem einfachen Betriebsmodus keinerlei Anfragen.

Mit Hilfe der Kommandozeilenoption `-B` lässt sich BOOTP deaktivieren und mit Hilfe der Kommandozeilenoption `-D` DHCP aktivieren. Mit Hilfe

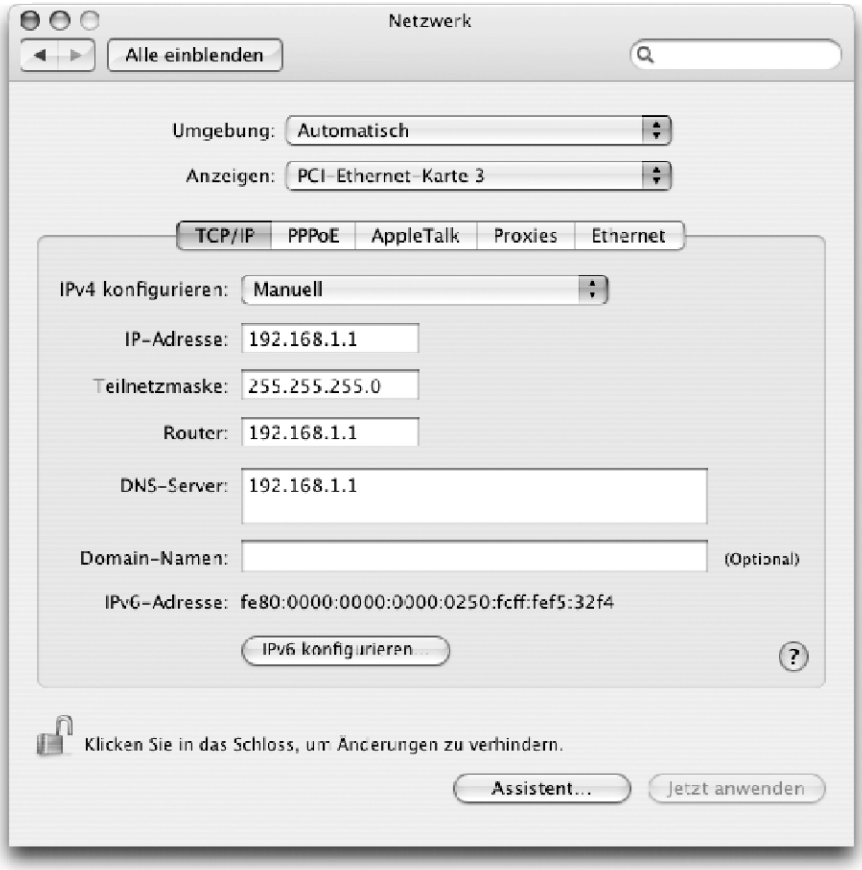


Abb. 3.17. Die Netzwerkschnittstelle des zukünftigen DHCP-Servers muss aktiv und konfiguriert sein. Insbesondere muss sie über eine gültige IP-Adresse verfügen.

der Option `-i` kann man die Netzwerkschnittstellen angeben, auf denen der DHCP-Server aktiviert werden soll.

```
$ sudo /usr/libexec/bootpd -d -B -D -i en1
Password:
bootpd[675]: interface en1: ip 192.168.1.1 mask 255.255.255.0
default: 192.168.0.1
127 ==> link 0
169.254 ==> link 4
192.168.0 ==> link 4
192.168.2 ==> link 5
...
```

Alle weiteren Einstellungen, also insbesondere die zu verteilenden Konfigurationsparameter, müssen in die lokale NetInfo-Datenbank eingetragen werden. Ohne diese Konfigurationsparameter werden eingehende DHCP-Anfragen lediglich mitprotokolliert, aber nicht beantwortet:

```
...
destination address 255.255.255.255
bootpd[675]: DHCP DISCOVER [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
no ip addresses
bootpd[675]: service time 0.000565 seconds
...
```

Der DHCP-Server liest die NetInfo-Konfigurationseinträge beim Start und nach Erhalt eines SIGHUP-Signals. Die Konfiguration des DHCP-Servers kann damit sehr einfach im laufenden Betrieb aktualisiert werden:

```
$ sudo killall -HUP bootpd
```

Die Konfigurationseinträge lassen sich am einfachsten mit dem Dienstprogramm *NetInfo Manager* erstellen.

Globale Einstellungen und Filter werden als Attribute (bzw. Eigenschaften) des NetInfo-Eintrags /config/dhcp definiert. Die Attribute `bootp_enabled` und `dhcp_enabled` enthalten eine Liste der Netzwerkschnittstellen, für die das jeweilige Protokoll aktiviert werden soll. Auf diese Weise kann man individuell festlegen, welches Protokoll auf welcher Netzwerkschnittstelle aktiviert und auf welcher nicht aktiviert werden soll. Um dann beispielsweise DHCP für die Netzwerkschnittstelle en1 zu aktivieren, und BOOTP für alle Schnittstellen zu deaktivieren, muss man das Attribut `bootp_enabled` mit einem leeren Wert und das Attribut `dhcp_enabled` mit dem Wert en1 versehen (Abb. 3.18).

Achtung: ein Protokoll wird über NetInfo deaktiviert, indem man das entsprechende Attribut mit einem leeren Wert versieht. Fehlt bei einem Attribut hingegen der Wert (Anzeige `<kein Wert>` im NetInfo-Manager), wird das jeweilige Protokoll für *alle* Netzwerkschnittstellen aktiviert.

Außer diesen grundlegenden Festlegungen bietet `bootpd` an dieser Stelle mit Hilfe einer Positivliste (Attribut `allow`) und einer Negativliste (Attribut `deny`) die Möglichkeit, anhand von Ethernetadressen zu definieren, welchen Hosts überhaupt geantwortet wird.

Wenn die Positivliste existiert, werden BOOTREQUESTs nur dann beantwortet, wenn die Ethernetadresse des Absenders in dieser Liste existiert.

Abb. 3.18. Die Attribute bootp_enabled und dhcp_enabled können jeweils eine Liste von Netzwerkschnittstellen enthalten, die für den jeweiligen Dienst verwendet werden sollen.

Wenn die Negativliste existiert, werden umgekehrt BOOTREQUESTs nur dann beantwortet, wenn die Ethernetadresse des Absenders *nicht* in der Negativliste steht. Auf diese Weise kann mit Hilfe der Positivliste die Wirkung des Servers auf ausgewählte, bekannte Hosts beschränkt werden – z. B. als eine (zugegebenermaßen sehr einfache) Sicherheitsmaßnahme. Umgekehrt kann mit Hilfe der Negativliste eine Gruppe von Hosts explizit vom Bezug der Konfigurationsparameter über BOOTP/DHCP ausgeschlossen werden – z. B. weil es sich um manuell zu konfigurierende Server-Rechner handelt.

Diese Mindestkonfiguration ist notwendig, aber für den Betrieb noch nicht hinreichend, da wir bisher nicht definiert haben, welche Informationen (also insbesondere welche IP-Adressen) verteilt werden sollen. Für diese Aufgabe müssen wir in der lokalen NetInfo-Datenbank einen neuen Eintrag unter `/config/dhcp/subnets` anlegen.

Dieser Eintrag beschreibt einen Pool von zu vergebenden IP-Adressen, zusammen mit den dazugehörigen Informationen wie Subnetz-Maske, Router-Adresse, DNS-Server-Adresse sowie optionalen weiteren Daten.

Zu den wichtigsten Attributen gehören:

- `name`: Eine frei wählbare Beschreibung des Eintrags, z. B. `Intern (Ethernet)`
- `net_address`: Die Adresse des IP-Netzwerks, z. B. `192.168.1.0`
- `net_mask`: Die Subnetz-Maske, z. B. `255.255.255.0`
- `net_range`: Ein Attribut mit zwei Werten: der erste Wert definiert die Startadresse und der zweite Wert die Endadresse eines IP-Adressen-Pools, also z. B. `192.168.1.2` und `192.168.1.254`[8]
- `client_types`: Enhält entweder den Wert `dhcp` oder den Wert `bootp` und legt fest, ob der Pool von DHCP oder von BOOTP verwendet werden soll.
- `lease_max`: Anzahl von Sekunden nach deren Ablauf eine DHCP-Adresse vom Empfänger „zurückzugeben“ ist, z. B. `3600` (eine Stunde).
- `dhcp_router`: IP-Adresse des Standardrouters, z. B. `192.168.1.1`
- `dhcp_domain_name_server`: IP-Adressen der DNS-Server. Falls dieses Attribut fehlt, verteilt der DHCP-Server einfach die aus der Konfiguration des eigenen DNS-Resolvers bekannten DNS-Server-Adressen.

Mit Hilfe dieser Attribute können wir einen voll funktionsfähigen DHCP-Server konfigurieren. Am einfachsten lassen sich die entsprechenden Eintragungen mit dem NetInfo-Manager vornehmen (siehe Abb. 3.19).

Anschließend muss man den DHCP-Server nur noch starten[9]:

```
$ sudo /usr/libexec/bootpd -d
bootpd[772]: interface en0: ip 192.168.0.10 mask 255.255.255.0
bootpd[772]: interface en1: ip 192.168.1.1 mask 255.255.255.0
default: 192.168.0.1
127 ==> link 0
169.254 ==> link 4
192.168.0 ==> link 4
192.168.1 ==> link 5
bootpd[772]: server name Tiger-Server.local
DNS 192.168.0.1
.: 1 entries:
Entry 0: Intern (Ethernet)
        address 192.168.1.0
        mask 255.255.255.0
        range 192.168.1.2..192.168.1.254
        client types:{ dhcp }
destination address 255.255.255.255
bootpd[772]: DHCP DISCOVER [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
max_lease is 3600
state=INIT

Sending: DHCP OFFER (size 300)
bootpd[772]: OFFER sent <no hostname> 192.168.1.2 pktsize 300
bootpd[772]: service time 0.002391 seconds
```

[8] Die Reihenfolge ist hierbei wichtig: der DHCP-Server meldet einen Konfigurationsfehler, falls die beiden Werte versehentlich vertauscht worden sind.

[9] Falls der DHCP-Server bereits läuft, genügt ein HUP-Signal, z. B. mit sudo killall -HUP bootpd, um die Konfigurationsdaten neu einzulesen, ohne den Prozess beenden und neu starten zu müssen.

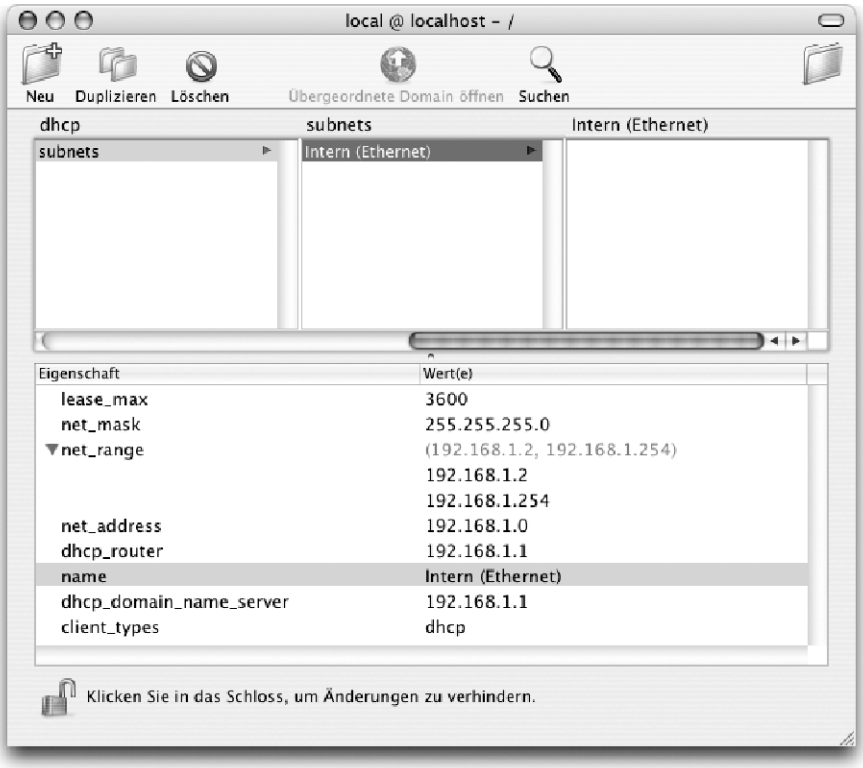


Abb. 3.19. Mit der Definition eines IP-Adressen-Pools wird die Konfiguration des DHCP-Servers vervollständigt.

```
destination address 255.255.255.255
bootpd[772]: DHCP REQUEST [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
SELECT
state=SELECT

Sending: DHCP ACK (size 300)
bootpd[772]: ACK sent Tiger-Client 192.168.1.2 pktsize 300
bootpd[772]: service time 0.003821 seconds
...
```

Im Beispiel ist erkennbar, dass der Netzwerkschnittstelle en1 die IP-Adresse 192.168.1.1 zugewiesen wurde. `bootpd` hat nach dem Start die in NetInfo eingetragenen Konfigurationsparameter übernommen. Etwas weiter unten kann man sehen, dass von einem Rechner mit der Ethernetadresse 00:30:65:a0:7b:be ein DHCP BOOTREQUEST empfangen worden ist und daraufhin die IP-Adresse 192.168.1.2 (also die erste IP-Adresse aus dem weiter oben definierten Pool) für eine Stunde (3600 Sekunden) zugeteilt wurde.

Die zugeteilten IP-Adressen werden in `/var/db/dhcpd_leases` persistent gespeichert. Auf diese Weise kann der DHCP-Server wiederholte Anfragen ein und desselben Hosts auch mit der gleichen Adresse beantworten:

```
$ cat /var/db/dhcpd_leases
{
        name=Tiger-Client
        ip_address=192.168.1.2
        hw_address=1,0:30:65:a0:7b:be
        identifier=1,0:30:65:a0:7b:be
        lease=0x42da79a4
}
```

Anfragen von neuen Hosts beantwortet der DHCP-Server einfach mit einer noch nicht vergebenen Adresse aus dem Pool. Ist der Pool ausgeschöpft (d. h. sind alle IP-Adressen aus dem Pool gerade an DHCP-Clients „verliehen“), werden weitere Anfragen ignoriert.

Wünscht man sich mehr Kontrolle darüber, welchem Host welche IP-Adresse zugeteilt werden soll, kann man auf die ebenfalls von `bootpd` unterstützte statische Adressenzuordnung zurückgreifen. Für jeden Host, dem nicht eine beliebige, sondern eine ganz bestimmte IP-Adresse zugeteilt werden soll, muss dazu ein Eintrag im Verzeichnis `/machines` der NetInfo-Datenbank erstellt werden.

Ein solcher Eintrag setzt sich (mindestens) aus den folgenden drei Attributen zusammen:

- `name`: Ein beliebiger Name (häufig der DNS-Name des Hosts), z. B. `iMac-G3`.
- `en_address`: Die Ethernetadresse des Hosts als eindeutiges Identifikationsmerkmal, z. B. `00:30:65:a0:7b:be`.
- `ip_address`: Die dem Host eindeutig zugewiesene IP-Adresse.

Hat man einen entsprechenden `/machines`-Eintrag in der NetInfo-Datenbank erzeugt, wird die in diesem Eintrag statisch (also fest) einem Host zugewiesene IP-Adresse automatisch aus dem IP-Adressenpool herausgenommen. Das bedeutet, dass diese IP-Adresse nur noch für den anhand der Ethernetadresse eindeutig identifizierten Host reserviert bleibt und ansonsten nicht verwendet wird.

Umgekehrt wird dem im Eintrag anhand der Ethernetadresse identifizierten Host statt der ersten IP-Adresse aus dem Pool die im Eintrag festgelegte IP-Adresse angeboten:

```
$ sudo /usr/libexec/bootpd -d
Password:
bootpd[789]: interface en0: ip 192.168.0.10 mask 255.255.255.0
bootpd[789]: interface en1: ip 192.168.1.1 mask 255.255.255.0
default: 192.168.0.1
127 ==> link 0
169.254 ==> link 4
192.168.0 ==> link 4
192.168.1 ==> link 5
```

```
bootpd[789]: server name Tiger-Server.local
DNS 192.168.0.1
.: 1 entries:
Entry 0: Intern (Ethernet)
        address 192.168.1.0
        mask 255.255.255.0
        range 192.168.1.2..192.168.1.254
        client types:{ dhcp }
destination address 255.255.255.255
bootpd[789]: DHCP REQUEST [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
max_lease is 3600
init-reboot
sending a NAK: 'requested address incorrect'
state=INIT/REBOOT

Sending: DHCP NAK (size 300)
bootpd[789]: NAK sent iMac-G3 192.168.1.42 pktsize 300
bootpd[789]: service time 0.002725 seconds
destination address 255.255.255.255
```

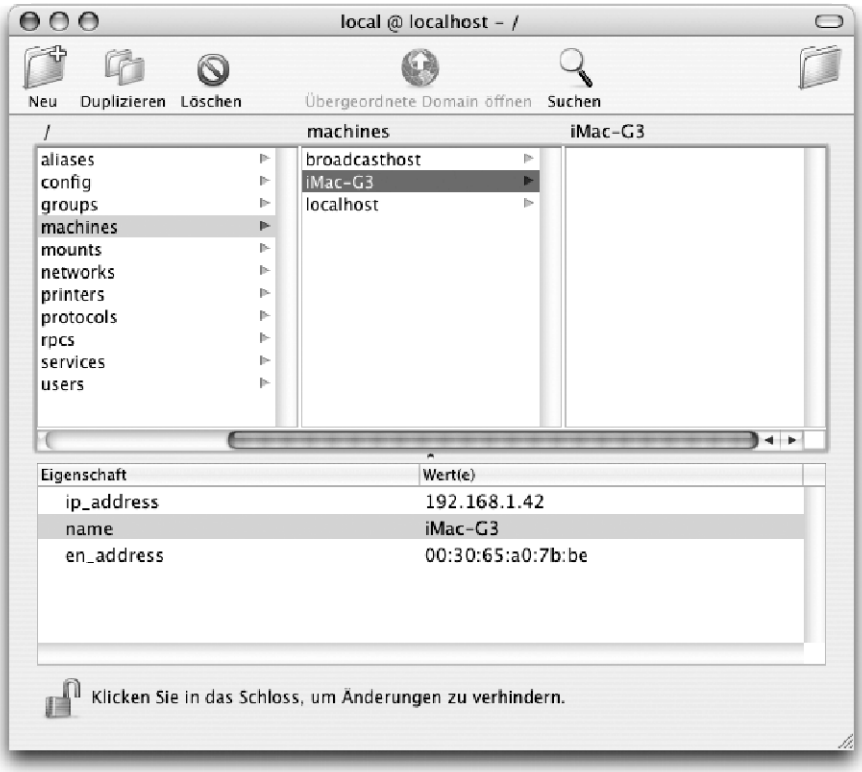


Abb. 3.20. Über Einträge im /machines-Verzeichnis der NetInfo-Datenbank unterstützt bootpd auch statisches DHCP. Die Einträge localhost und broadcasthost haben für DHCP keine Bedeutung, sondern haben eine ähnliche Funktion wie die entsprechenden Einträge in /etc/hosts auf anderen Plattformen.

```
bootpd[789]: DHCP DISCOVER [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
max_lease is 3600
state=INIT

Sending: DHCP OFFER (size 300)
bootpd[789]: OFFER sent iMac-G3 192.168.1.42 pktsize 300
bootpd[789]: service time 0.002430 seconds
destination address 255.255.255.255
bootpd[789]: DHCP REQUEST [en1]: 1,0:30:65:a0:7b:be <Tiger-Client>
max_lease is 3600
SELECT
state=SELECT

Sending: DHCP ACK (size 300)
bootpd[789]: ACK sent iMac-G3 192.168.1.42 pktsize 300
bootpd[789]: service time 0.001923 seconds
...
```

Im Beispiel sieht man, dass dem anhand der Ethernetadresse identifizierten DHCP-Client die weitere Verwendung seiner bisherigen IP-Adresse sogar verweigert wurde (*sending a NAK: 'requested address incorrect'*). Der DHCP-Server hat die Verlängerung der Leihfrist verweigert, die bisherige IP-Adresse „zurückgenommen“ und die neue, statisch zugewiesene IP-Adresse zugeteilt.

Der `/machines`-Eintrag ersetzt nicht, sondern ergänzt den `/subnets`-Eintrag. Wie man an unserem Beispiel sieht, fehlen im `/machines`-Eintrag Angaben zu Netzwerk-Maske, Router-Adresse und Leihfrist. Diese Angaben werden vom Server automatisch vervollständigt: Dazu prüft er einfach, in welchen Pool die im `/machines`-Eintrag festgelegte IP-Adresse fallen würde, und ergänzt seine Antwort um die dort festgelegten Angaben.

In der Praxis wird man `bootpd`, falls man mit ihm nicht nur experimentieren, sondern ihn auch richtig einsetzen möchte, nicht immer wieder manuell wie oben beschrieben starten wollen. Um den DHCP-Server nach einem Systemstart automatisch zu aktivieren, kann man unter Mac OS X 10.4 die bereits existierende Launchd-Konfigurationsdatei verwenden (Abb. 3.21):

```
$ sudo launchctl load -w \
> /System/Library/LaunchDaemons/bootps.plist
```

In Mac OS X 10.3 ist eine entsprechende `xinetd`-Konfigurationsdatei vorhanden, die ebenfalls nur noch aktiviert werden muss:

```
$ cat /etc/xinetd.d/bootps
service bootps
{
        disable         = yes
        socket_type     = dgram
        wait            = yes
        user            = root
        server          = /usr/libexec/bootpd
        groups          = yes
        flags           = REUSE
}
```

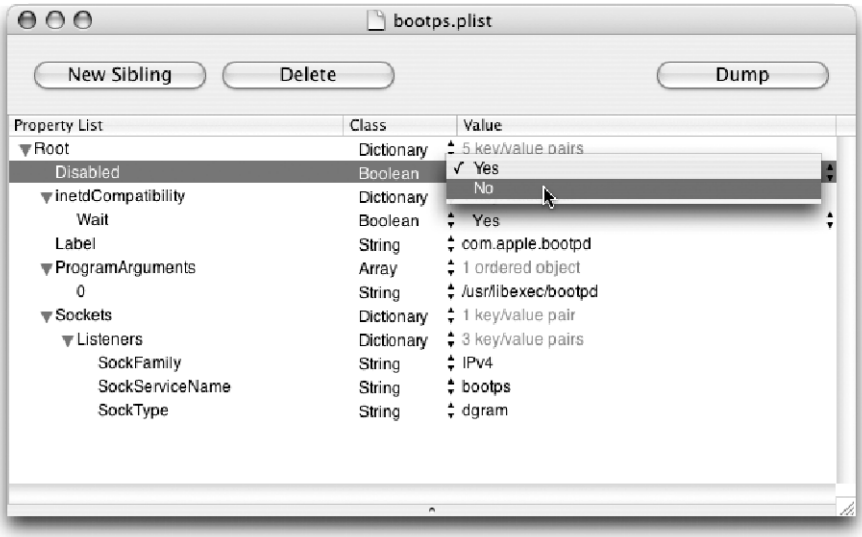


Abb. 3.21. Eine Launchd-Konfigurationsdatei für den DHCP-Server existiert unter Mac OS X 10.4 bereits.

Man ändert dazu lediglich die Zeile `disable = yes` in `disable = no`. Damit `xinetd` auch ohne Systemneustart die geänderte Konfigurationsdatei einliest, kann man ihm ein SIGHUP-Signal schicken:

```
$ sudo kill -HUP ‘cat /var/run/xinetd.pid‘
```

Sowohl unter Mac OS X 10.4 als auch unter Mac OS X 10.3 sollte das Skript `/sbin/service` den Start des DHCP-Servers erledigen:

```
$ sudo service bootps start
```

Benutzer älterer Mac-OS-X-Versionen müssen hingegen ein Startobjekt (Startup Item) erstellen.

3.4.2 Firewall

Als Firewall (Brandschutzmauer) bezeichnet man einen Host, der die Kommunikation zwischen einem internen Netzwerk und dem öffentlichen Internet einschränkt. Häufig geschieht dies aus Sicherheitsgründen, nämlich um zu verhindern, dass Fremde Zugang zu internen Resourcen und Informationen bekommen. Umgekehrt können Verfahrens- oder rechtliche Gründe die Beschränkung des Zugangs zu bestimmten externen Resourcen diktieren, wie z. B. zu Online-Tauschbörsen, bestimmten Web-Sites u.ä.

Grundsätzlich gibt es zwei Möglichkeiten, den Zugang zu beschränken: durch einen Stellvertreter-Host (Proxy), oder durch einen Paketfilter. Beide

Methoden erfordern einen Host, der über mindestens zwei Netzwerkschnittstellen verfügt und sowohl an das interne als auch an das externe Netzwerk angeschlossen ist.

Ein Stellvertreter-Host leitet im Gegensatz zu einem normalen Gateway (und einem Paketfilter-Host) keine Pakete weiter. Stattdessen beherbergt er Stellvertreterprogramme, die den Zugriff auf bestimmte Dienste im externen Netzwerk erlauben. Umgekehrt können externe Hosts nur auf den Stellvertreterhost und die dort verfügbaren Dienste und nicht auf die internen Hosts zugreifen (siehe dazu auch Abschn. 2.1.4).

Ein Paketfilter-Host wird dagegen als Router konfiguriert und leitet Pakete zwischen den Netzwerkschnittstellen hin und her. Im Gegensatz zu einem normalen Router vergleicht ein Paketfilter-Host jedes Paket vor der Weiterleitung mit einem Satz von Regeln, mit deren Hilfe der Administrator festlegen kann, welche Pakete weitergeleitet und welche Pakete verworfen werden sollen.

Ein Paketfilter kann manchmal auch auf einem Host mit nur einer Netzwerkschnittstelle nützlich sein. Mit entsprechenden Regeln lassen sich z. B. Zugriffe von außen auf nur intern benötigte Dienste verhindern oder der Nutzerkreis von bestimmten Diensten auf einzelne, vertrauenswürdige Netzwerke beschränken.

Die Paketfilter-Funktion wird unter Mac OS X durch die `ipfw`-Software realisiert. Der eigentliche Paketfilter sitzt dabei im Kernel und kann über die Zustandsvariable `net.inet.ip.fw.enable` ein- und ausgeschaltet werden. Unter Mac OS X 10.4 ist der Paketfilter im Kernel normalerweise immer aktiv.

```
$ sysctl net.inet.ip.fw.enable
net.inet.ip.fw.enable: 1
$ sudo sysctl -w net.inet.ip.fw.enable=0
net.inet.ip.fw.enable: 1 -> 0
$ sudo sysctl -w net.inet.ip.fw.enable=1
net.inet.ip.fw.enable: 0 -> 1
```

Die Steuerung des Paketfilters kann entweder (sehr eingeschränkt) über den Bereich *Firewall* in der Systemeinstellung Sharing, oder (im vollen Umfang) über den Befehl `ipfw` erfolgen.

In jedem Fall erfolgt die Steuerung mit Hilfe einer Liste von nummerierten Regeln. Für jedes Paket geht der Paketfilter eine Regelliste Regel für Regel durch und prüft die Anwendbarkeit jeder einzelnen Regel. Ist eine Regel anwendbar, führt der Paketfilter die in der Regel definierte Aktion aus und macht je nach Aktion entweder mit der nächsten Regel weiter oder bricht die Bearbeitung ab.

Die Regeln werden einfach von 1 bis 65535 durchnummeriert, wobei die Nummer 1 der Regel mit der höchsten und 65535 der Regel mit der niedrigsten Priorität entspricht. Die Regelliste besteht immer aus mindestens einer Regel. Diese Standardregel hat die niedrigstmögliche Priorität 65535 und kann vom Benutzer nicht geändert werden. Sie legt das Standardverhalten

des Paketfilters für den Fall fest, dass keine weiteren Regeln definiert worden sind. In Mac OS X stellt diese Regel sicher, dass der Paketfilter im Auslieferungszustand offen ist, d. h. keine Pakete filtert bzw. blockiert.

Die aktuell gültige Liste von Regeln kann jederzeit mit dem Befehl `ipfw` angezeigt werden:

```
$ sudo ipfw list
65535 allow ip from any to any
```

Im Auslieferungszustand wird natürlich nur die Standardregel angezeigt. Sie beginnt mit dem Index (hier `65535`), gefolgt von Aktion (hier `allow`, also erlauben), Protokoll (hier `ip`), Absender (hier `any`, also jeder Absender) und Empfänger (hier auch `any`, also jeder Empfänger):

```
65535 allow ip from any to any
```

Anhand des Protokolls, des Absenders und des Empfängers (zusammen mit eventuell vorhandenen Angaben zur Schnittstelle und weiteren Optionen) entscheidet der Paketfilter, ob die Regel auf ein bestimmtes Paket angewendet werden kann. Die Standardregel gilt für alle Pakete unabhängig vom verwendeten Protokoll (ip ist ein Synonym für all, siehe weiter unten), und unabhängig von Empfänger (any) und Absender (auch any).

Allgemein kann eine Regel folgendes Aussehen annehmen:

index [**prob** *wahrscheinlichkeit*] *aktion* [**log** [**logamount** *anzahl*]] *protokoll* **from** *absender* **to** *empfänger* [*schnittstelle*] [*option(en)*]

Jede Regel beginnt mit ihrem Index. Optional folgt danach die Wahrscheinlichkeit (ausgedrückt als eine Zahl zwischen 0 und 1), mit der IPFW versucht, die Regel anzuwenden. Diese Option kann dazu verwendet werden, Paketverluste zu simulieren, findet aber selten Anwendung.

Von zentraler Bedeutung ist die Angabe der Aktion, also die Entscheidung, was mit einem Paket passieren soll, falls die Regel angewendet werden kann. Die Aktion impliziert auch jeweils die Fortsetzung oder (bei den meisten Aktionen) den Abbruch der weiteren Regelüberprüfung. Die wichtigsten Aktionen lauten:

allow Paket darf passieren. Die Regelüberprüfung wird beendet.

pass, **permit** und **accept** Synonyme für **allow**.

deny Paket darf nicht passieren und wird blockiert bzw. gefiltert. Die Regelüberprüfung wird beendet.

drop Synonym für **deny**.

unreach *code* Ähnlich wie **deny**. IPFW verschickt in diesem Fall allerdings zusätzlich noch eine ICMP-Fehlermeldung (Destination Unreachable) an den Absender. Über den Parameter *code* kann der genaue Untertyp der Fehlermeldung spezifiziert werden. Der Untertyp kann entweder numerisch oder symbolisch angegeben werden. Erlaubt sind die 16 Symbole **net**, **host**, **protocol**, **port**, **needfrag**, **srcfail**, **net-unknown**,

host-unknown, **isolated**, **net-prohib**, **host-prohib**, **tosnet**, **toshost**, **filter-prohib**, **host-precedenc** und **precedence-cutoff**. Die Symbole entsprechen den Untertypen 0 bis 15 der ICMP-Meldung Destination Unreachable. Die Regelüberprüfung wird beendet.

reject Entspricht **unreach host** (Host unreachable). In der man-Page als „deprecated“ markiert und daher nicht mehr empfehlenswert. Die Regelüberprüfung wird beendet.

reset Ähnlich wie **deny**, jedoch nur für TCP-Pakete. IPFW verwirft das betroffene Paket und schickt dem Absender ein TCP RST-Paket um den Abbruch der TCP-Verbindung zu signalisieren. Die Regelüberprüfung wird beendet.

skipto *nr* Weist IPFW an, die Regelüberprüfung mit Regel Nummer *nr* fortzusetzen.

Über das Schlüsselwort **log** kann für jede Regel optional die Protokollierung aktiviert werden. Dabei muss zuvor die Protokollfähigkeit des Paketfilters über die Zustandsvariable net.inet.ip.fw.verbose im Kernel aktiviert werden, da diese im Auslieferungszustand von Mac OS X deaktiviert ist:

```
$ sysctl net.inet.ip.fw.verbose
net.inet.ip.fw.verbose: 0
$ sudo sysctl -w net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose: 0 -> 1
```

Aktiviert man für eine bestimmte Regel die Protokollierung, erzeugt der Paketfilter jedesmal, wenn die Regel zur Anwendung kommt (also bei jedem passenden Paket) einen Eintrag in /var/log/system.log[10]. Da auf diese Weise die Verarbeitungsgeschwindigkeit des verwendeten Rechners leiden und auch die Protokolldatei sehr schnell eine beachtliche Größe erreichen könnte, kann die Anzahl der Protokolleinträge pro Regel mit der Angabe von **logamount** begrenzt werden.

Fehlt diese Angabe, verwendet der Paketfilter den in der Zustandsvariablen `net.inet.ip.fw.verbose_limit` eingestellten Wert – im Auslieferungszustand von Mac OS X 10.4 den Wert 0, der für keinerlei Begrenzung steht.

Der Paketfilter zählt die Protokolleinträge für jede Regel mit Hilfe eines Zählers. Falls der Zähler die maximale Anzahl der Protokolleinträge für eine Regel erreicht, wird die Protokollierung der Regel ausgesetzt. Um die Protokollierung fortzusetzen, muss man den Protokollzähler unter Angabe der Regelnummer zurücksetzen. Lautet der Index der betroffenen Regel z. B. 10, kann der Zähler mit dem folgenden Befehl zurückgesetzt werden:

```
$ sudo ipfw resetlog 10
Entry 10 logging count reset
```

Während die Angaben zur Protokollierung optional sind, muss jede Regel die danach folgenden Angaben zum Protokoll, dem Absender und dem

[10] Da die Protokollierung über syslogd erfolgt, kann das Protokoll auch in eine andere Datei umgelenkt werden.

Empfänger enthalten. Auf der Basis dieser Angaben entscheidet der Paketfilter, ob ein Paket zur Regel „passt“.

Als Protokoll kann dabei jedes in `/etc/protocols` gelistete Protokoll angegeben werden, also insbesondere **tcp** und **udp**. Mit der Angabe eines bestimmten Protokolls gilt die betreffende Regel nur noch für IP-Pakete, die als Träger für das angegebene Protokoll dienen. Alternativ können die Schlüsselwörter **ip** oder **all** verwendet werden, wenn keine Einschränkungen im Bezug auf das verwendete Protokoll gemacht werden sollen.

Als Absender und Empfänger sind folgende Angaben zulässig:

any Jede beliebige IP-Adresse.
me Die IP-Adressen der eigenen Netzwerkschnittstellen.
[**not**] *ipadresse* Eine bestimmte IP-Adresse in der üblichen Dezimalschreibweise. Mit **not** alle IP-Adressen *außer* der angegebenen.
[**not**] *netzadresse*/*bits* Ein bestimmtes Netzwerk mit der Netzwerkadresse *netzadresse* und Netzmaske der Länge *bits*, z. B.: 192.168.0.0/24 (192.168.0.0. bis 192.168.0.255). Mit **not** alle IP-Adressen *außer* der im angegebenen Netzwerk.
[**not**] *netzadresse*:*maske* Ein bestimmtes Netzwerk mit der Netzwerkadresse *netzadresse* und Netzmaske *maske*, z. B.: 192.168.0.0:255.255.255.0 (ebenfalls 192.168.0.0 bis 192.168.0.255). Mit **not** alle IP-Adressen *außer* der im angegebenen Netzwerk.

Bei Regeln, welche die Protokolle TCP oder UDP betreffen, können die Angaben zu Absender und Empfänger durch einzelne Ports, Portbereiche oder Portlisten ergänzt werden. Die Angaben zu Ports können entweder numerisch oder symbolisch (entsprechend den Einträgen in /etc/services) erfolgen (Beispiele dazu weiter unten).

Optional kann die Anwendbarkeit einer Regel auch auf eine bestimmte Datenflussrichtung oder eine bestimmte Netzwerkschnittstelle eingeschränkt werden. Dazu können folgende Schnittstellenangaben kombiniert werden:

in Alle eintreffenden Pakete.
out Alle ausgehenden Pakete.
via *schnittstelle* Alle über die Netzwerkschnittstelle *schnittstelle* übertragenen Pakete. Die Schnittstelle wird symbolisch – z. B. als en0 oder en1 – angegeben.
via *schnittstellen** Alle über die Schnittstellenfamilie *schnittstellen* übertragenen Pakete. Die Schnittstellenfamilie wird symbolisch über das gemeinsame Präfix gefolgt von einem Stern – z. B. en* – angegeben.
via any Alle über irgendeine Netzwerkschnittstelle übertragenen Pakete (Standard, auch wenn diese Angabe fehlt).
via *ipadresse* Alle über die Netzwerkschnittstelle mit der IP-Adresse *ipadresse* übertragenen Pakete.
recv *schnittstelle* Alle über die Netzwerkschnittstelle *schnittstelle* empfangenen Pakete.

xmit *schnittstelle* Alle über die Netzwerkschnittstelle *schnittstelle* gesendeten Pakete (nur zusammen mit **out** zulässig).

Die Regelspezifikation endet mit einer oder mehreren Optionen, welche eine noch feinere Spezifikation der mit der Regel zu erfassenden Pakete erlauben. Beispiele sind:

setup Pakete, die den Aufbau einer TCP-Verbindung initiieren (gesetztes SYN-Flag, aber kein ACK-Flag).

established Pakete, die zu einer bereits aufgebauten TCP-Verbindung gehören (gesetztes ACK-Flag oder RST-Flag).

icmptypes *typenliste* ICMP-Pakete eines bestimmten Typs. Über diese Option lassen sich z. B. Antworten auf pings (ICMP Echo Reply, Typ 0) unterdrücken.

Die einfachste Möglichkeit, neue Regeln hinzuzufügen, bietet sich über Systemeinstellungen Sharing (Abb. 3.22). Ein Klick auf den Firewall-Startknopf aktiviert eine Reihe von Regeln, was man über `ipfw` leicht nachprüfen kann:

```
$ sudo ipfw list
02000 allow ip from any to any via lo*
02010 deny ip from 127.0.0.0/8 to any in
02020 deny ip from any to 127.0.0.0/8 in
02030 deny ip from 224.0.0.0/3 to any in
02040 deny tcp from any to 224.0.0.0/3 in
02050 allow tcp from any to any out
02060 allow tcp from any to any established
12190 deny tcp from any to any
65535 allow ip from any to any
```

Die erste Regel (Nummer 2000) lässt grundsätzlich alle Pakete durch, die über die Loopback-Schnittstelle, also Host-intern, verschickt werden:

```
02000 allow ip from any to any via lo*
```

Der Paketfilter bearbeitet in diesem Fall nur diese erste Regel, stellt fest, dass das Paket passieren kann und schließt die Bearbeitung eines solchen Loopback-Pakets damit sofort ab. Handelt es sich um ein Paket, welches über eine „echte“ Netzwerkschnittstelle übertragen wurde, müssen erst die weiteren Regeln geprüft werden, da die erste Regel für diesen Fall nicht anwendbar ist.

Die nächsten vier Regeln (2010, 2020, 2030 und 2040) behandeln ankommende Pakete mit ungültigen Absender- und Empfänger-Adressen. Die Regel 2010 blockiert alle Pakete mit einer Loopback-Absenderadresse (127.0.0.0 bis 127.255.255.255), die Regel 2020 diejenigen mit einer Loopback-Empfängeradresse:

```
02010 deny ip from 127.0.0.0/8 to any in

02020 deny ip from any to 127.0.0.0/8 in
```

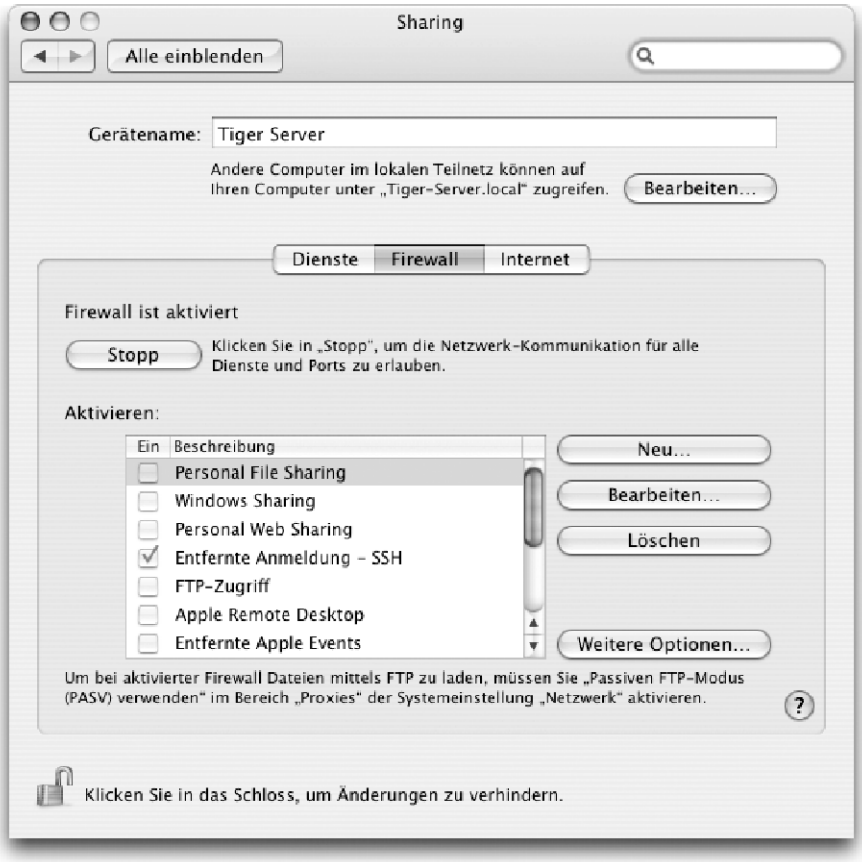


Abb. 3.22. Die einfachste Möglichkeit, den Paketfilter IPFW zu konfigurieren, bietet sich über Systemeinstellungen Sharing.

Die Regel 2030 blockiert alle Pakete mit Absenderadressen größer oder gleich 224.0.0.0, d. h. mit Absenderadressen aus dem Multicast-Bereich (Klasse D: 224.0.0.0 bis 239.255.255.255), aus dem reservierten Bereich (Klasse E: 240.0.0.0 bis 247.255.255.255), aus dem ungültigen Bereich oberhalb von 247.255.255.255 (248.0.0.0 bis 255.255.255.254) – inklusive der Broadcastadresse 255.255.255.255:

```
02030 deny ip from 224.0.0.0/3 to any in
```

Die Regel 2040 blockiert alle TCP-Pakete mit einer solchen ungültigen Empfängeradresse. UDP-Pakete sind von dieser Regel nicht betroffen und kommen natürlich durch – das ist auch sinnvoll, da UDP-Pakete sowohl an IP-Adressen aus dem Multicast-Bereich als auch an die Broadcastadresse geschickt werden dürfen.

```
02040 deny tcp from any to 224.0.0.0/3 in
```

Die Regeln 2050 und 2060 sorgen dafür, dass ausgehende TCP-Verbindungen nicht verhindert werden. Die Regel 2050 lässt dazu alle ausgehenden TCP-Segmente passieren:

```
02050 allow tcp from any to any out
```

Die Regel 2060 erlaubt ankommende TCP-Segmente, solange sie zu einer bereits bestehenden Verbindung gehören:

```
02060 allow tcp from any to any established
```

Die Regel 12190 blockiert alle anderen, also die noch nicht durch die bisherigen Regeln behandelten TCP-Pakete. Damit wird effektiv verhindert, dass TCP-Verbindungen von außen initiiert werden können.

```
12190 deny tcp from any to any
```

Den Abschluss bildet schließlich die Standardregel, die alle sonstigen, also insbesondere alle UDP-Pakete, durchlässt:

```
65535 allow ip from any to any
```

Interessanterweise werden UDP-Pakete im Auslieferungszustand des Paketfilters also nicht blockiert. Offenbar sieht Apple hier ein geringeres Sicherheitsrisiko als bei TCP. Im Gegensatz zu früheren Versionen lässt sich die Filterung von UDP-Paketen in Mac OS X 10.4 allerdings über die Schaltfläche „Weitere Optionen...“ zusätzlich aktivieren (Abb. 3.23).

Die Aktivierung der Option „UDP-Verkehr blockieren“ führt zu einer Reihe von neuen Regeln, die in der Systemeinstellung zwar nicht angezeigt werden, aber in der Kommendozeile sichtbar gemacht werden können:

```
$ sudo ipfw list
...
12190 deny tcp from any to any
20310 allow udp from any to any dst-port 53 in
20320 allow udp from any to any dst-port 68 in
20321 allow udp from any 67 to me in
20322 allow udp from any 5353 to me in
20340 allow udp from any to any dst-port 137 in
20350 allow udp from any to any dst-port 427 in
20360 allow udp from any to any dst-port 631 in
20370 allow udp from any to any dst-port 5353 in
22000 allow udp from any to any dst-port 123 in
30510 allow udp from me to any out keep-state
30520 allow udp from any to any in frag
35000 deny udp from any to any in
65535 allow ip from any to any
```

Wie man sehen kann, wurden durch die Aktivierung der Option „UDP-Verkehr blockieren“ zwölf neue Regeln zwischen Regel 12190 und Regel 65535 zusätzlich eingefügt.

Der neue Regelblock aus elf Regeln, die explizit bestimmte Arten von Paketen erlauben, und einer Abschlussregel, hier Regel Nummer 35000, die alle sonstigen UDP-Pakete blockiert:

```
35000 deny udp from any to any in
```

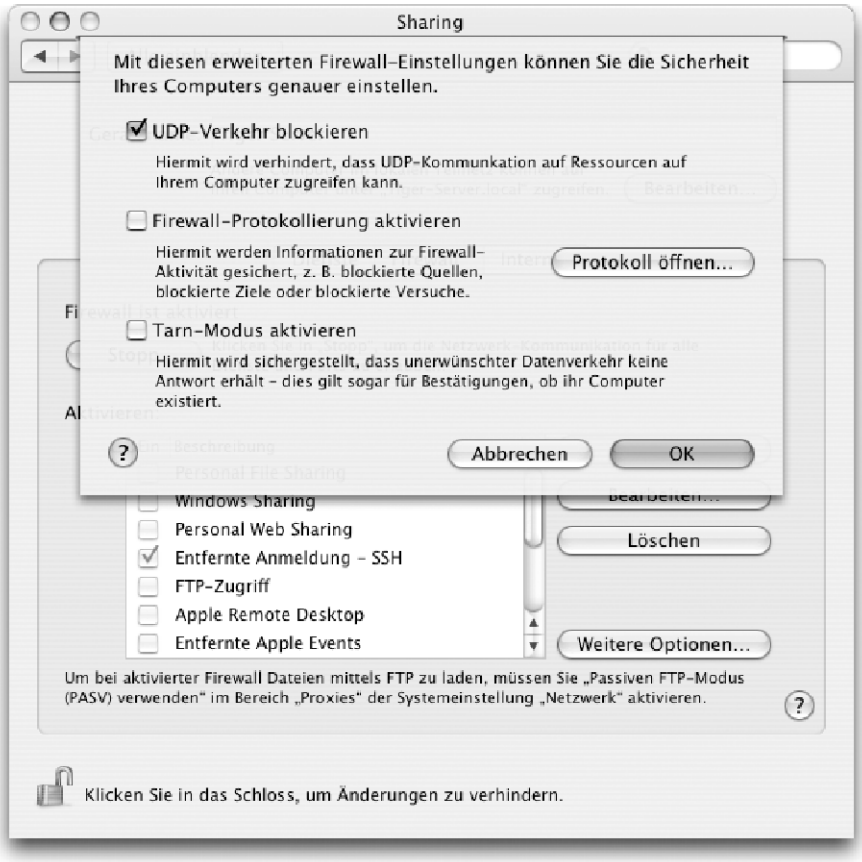


Abb. 3.23. Die Filterung von UDP-Paketen kann in Mac OS X 10.4 zusätzlich aktiviert werden.

Die Regeln 20310 bis 22000 sorgen dafür, das einige UDP-basierte Dienste trotz eingeschalteter Firewall erreichbar bleiben. Die ausgenommenen Dienste lassen sich auf der Basis der in den Regeln verwendeten Port-Nummern mit Hilfe der Datei /etc/services leicht identifizieren:

```
$ grep -E \
> '[[:space:]]53/udp|'\
> '[[:space:]]67/udp|'\
> '[[:space:]]68/udp|'\
> '[[:space:]]123/udp|'\
> '[[:space:]]137/udp|'\
> '[[:space:]]427/udp|'\
> '[[:space:]]631/udp|'\
> '[[:space:]]5353/udp' \
> /etc/services
domain          53/udp    # Domain Name Server
bootps          67/udp    # Bootstrap Protocol Server
```

```
bootpc          68/udp    # Bootstrap Protocol Client
ntp            123/udp    # Network Time Protocol
netbios-ns     137/udp    # NETBIOS Name Service
svrloc         427/udp    # Server Location
ipp            631/udp    # IPP (Internet Printing Protocol)
mdns          5353/udp    # Multicast DNS
```

Regel 20310 stellt demnach die Erreichbarkeit eines lokal betriebenen DNS-Servers sicher:

```
20310 allow udp from any to any dst-port 53 in
```

Die Regeln 20320 und 20321 sind für die korrekte Funktionsweise des eingebauten DHCP-Clients notwendig (für den Betrieb eines DHCP-Servers sind diese Regeln zu restriktiv):

```
20320 allow udp from any to any dst-port 68 in
```

```
20321 allow udp from any 67 to me in
```

Die Regeln 20322 und 20370 ermöglichen dem mDNS-Responder die Kommunikation mit der Außenwelt und sind daher für den Korrekten Betrieb von Bonjour notwendig:

```
20322 allow udp from any 5353 to me in
```

```
20370 allow udp from any to any dst-port 5353 in
```

Regel 20340 stellt sicher, dass nach der Aktivierung von Windows Sharing in der Systemeinstellung Sharing (bzw. der manuellen Aktivierung von Samba) der Rechner mit Hilfe seines NetBIOS-Namens trotz Firewall gefunden werden kann:

```
20340 allow udp from any to any dst-port 137 in
```

Regel 20350 schaltet den UDP-Port für das Service Location Protocol (SLP) frei. Dieses kann zusammen mit Personal File Sharing (bzw. dem AFP-Fileserver) verwendet werden:

```
20350 allow udp from any to any dst-port 427 in
```

Regel 20360 erlaubt schließlich dem Mac-OS-X-Druckerverwaltungssystem CUPS Informationen über im Netzwerk freigegebene Drucker zu erhalten:

```
20360 allow udp from any to any dst-port 631 in
```

Regel 30510 und Regel 30520 haben hingegen einen etwas anderen Charakter, indem sie sich nicht auf bestimmte Protokolle oder Dienste beziehen. Stattdessen erlaubt Regel 30510 grundsätzlich das Absenden von beliebigen UDP-Paketen und ist damit mit der auf ausgehenden TCP-Verkehr bezogenen Regel 3050 vergleichbar:

```
30510 allow udp from me to any out keep-state
```

Interessant ist dabei die bisher vernachlässigte Option `keep-state`. Mit dieser Option wird bei jeder Anwendung dieser Regel automatisch eine Regel erzeugt, die zukünftige Pakete mit dem gleichen Absender/Empfänger-Paar und dem gleichen Protokoll erfasst. Die Lebensdauer einer solchen *dynamischen* Regel ist begrenzt. Sie wird gelöscht, wenn längere Zeit keine passenden Pakete empfangen werden. Ein Kernelparameter legt dabei die maximale Wartezeit in Sekunden fest:

```
$ sysctl net.inet.ip.fw.dyn_udp_lifetime
net.inet.ip.fw.dyn_udp_lifetime: 10
```

Regel 30520 lässt hingegen Fragmente von UDP-Paketen passieren. Fragmente haben keinen gültigen Header und müssen daher separat behandelt werden:

```
30520 allow udp from any to any in frag
```

Weitere Regeln werden automatisch aktiviert, sobald man in Systemeinstellungen Sharing bei aktivierter Firewall einen der Standarddienste startet. So wird beim Start von Personal File Sharing die Regelliste automatisch um zwei Regeln erweitert:

```
$ sudo ipfw list
02000 allow ip from any to any via lo*
02010 deny ip from 127.0.0.0/8 to any in
02020 deny ip from any to 127.0.0.0/8 in
02030 deny ip from 224.0.0.0/3 to any in
02040 deny tcp from any to 224.0.0.0/3 in
02050 allow tcp from any to any out
02060 allow tcp from any to any established
02070 allow tcp from any to any 548 in
02080 allow tcp from any to any 427 in
12190 deny tcp from any to any
...
65535 allow ip from any to any
```

Die Regel 2070 erlaubt eingehende TCP-Verbindungen von außen auf den Port 548 (Apple File Sharing Protocol) und ermöglicht damit externen Hosts den Zugriff auf die mittels Personal File Sharing freigegebenen Verzeichnisse. Die Regel 2080 erlaubt eingehende TCP-Verbindungen auf den Port 427 (Service Location Protocol, kurz SLP) – diese sind für die Nutzung von Personal File Sharing nicht zwingend notwendig, ermöglichen aber SLP-kompatiblen Clients das einfachere Auffinden des Dienstes[11].

Die Systemeinstellungen Sharing bieten darüber hinaus die Möglichkeit, einfache zusätzliche Regeln zu definieren. Über den Knopf „Neu. . . “ gelangt man zu einem Sheet (Abb. 3.24), über das man eine Dienstbezeichnung und die dazugehörigen Portnummern angeben kann. Dabei hat man die Möglichkeit, die Dienstbezeichnung aus einer (kurzen) Liste von vordefinierten Diensten zu wählen (was einem die Eingabe der dazu passenden Portnummern erspart) oder die Dienstbezeichnung und die Portnummern frei zu wählen.

[11] Im Gegensatz zu SLP ist hier für Bonjour keine zusätzliche Regel erforderlich, da Bonjour bzw Multicast DNS ausschließlich UDP-Pakete verwendet.

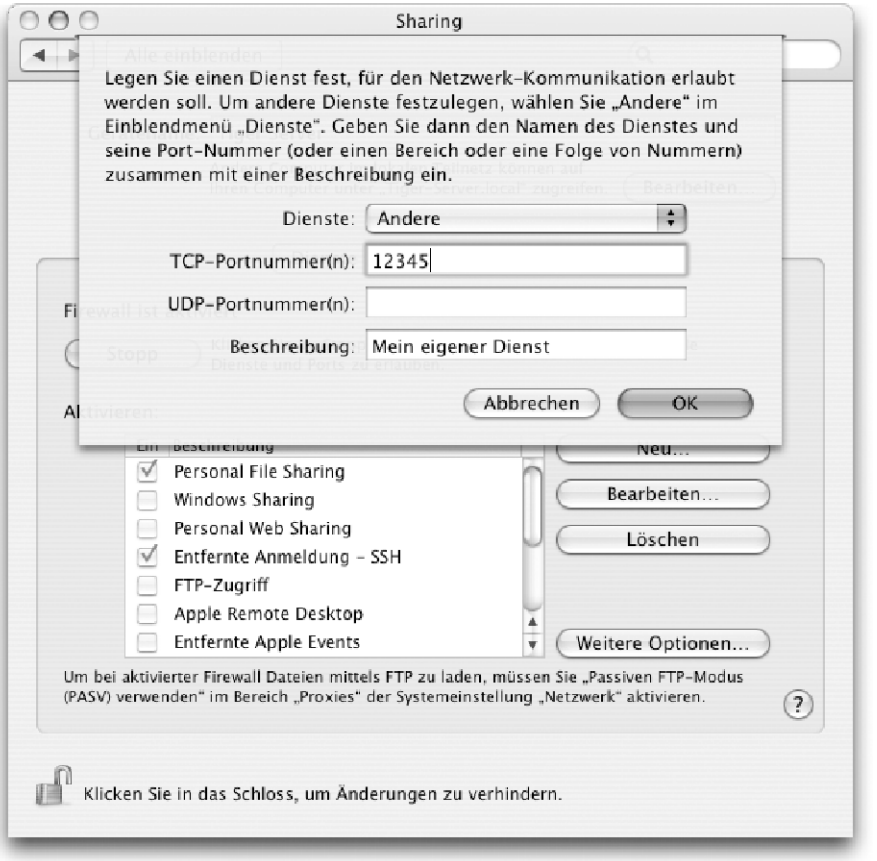


Abb. 3.24. Einfache Regeln lassen sich auch über die Systemeinstellungen definieren.

Eine auf diese Weise definierte Regel kann sowohl eingehende TCP-Verbindungen als auch eingehende UDP-Pakete betreffen:

```
$ sudo ipfw list
...
02090 allow tcp from any to any 12345 in
...
22000 allow udp from any to any dst-port 12345 in
...
65535 allow ip from any to any
```

Neben der Möglichkeit nicht nur TCP-Verbindungen, sondern auch UDP-Pakete zu filtern, wurde die graphische Benutzeroberfläche der Firewall in der Systemeinstellung Sharing in Mac OS X 10.4 um zwei weitere Funktionen ergänzt: der Benutzer kann die Protokollierung von gefilterten Paketen aktivieren und Pings blockieren (Abb. 3.25).

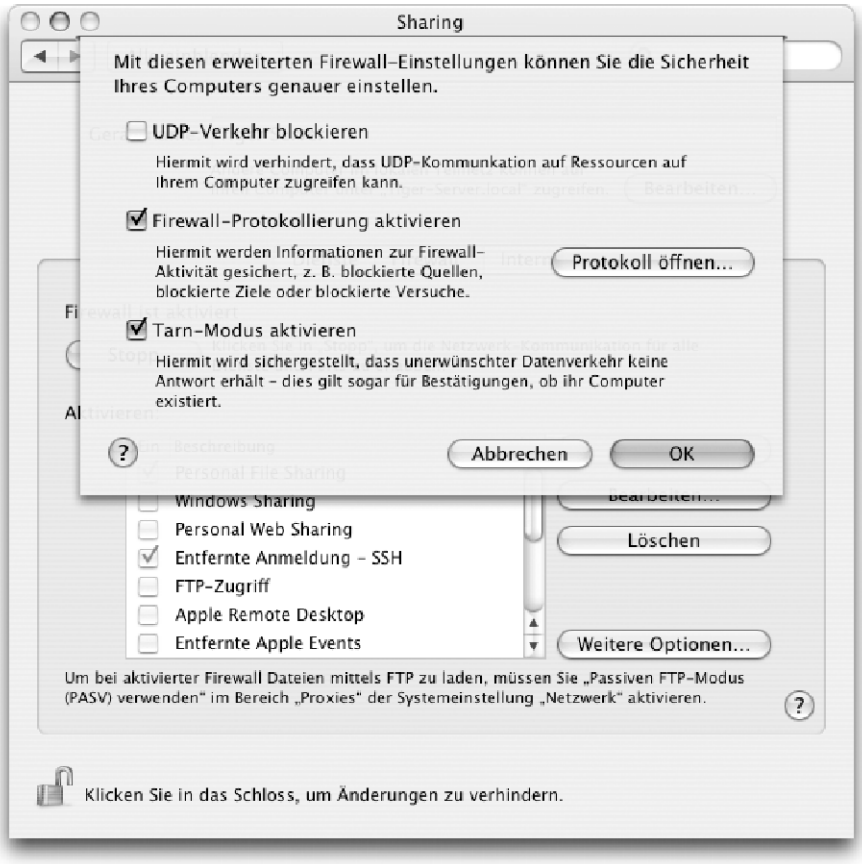


Abb. 3.25. In Mac OS X 10.4 lässt sich die Protokollierung von gefilterten Paketen und das Blockieren von Pings auf Knopfdruck einschalten.

Über die Option „Firewall-Protokollierung aktivieren“ kann man die Protokollierung aller gefilterten Pakete in die Datei /var/log/ipfw.log aktivieren. Die Option ergänzt die beiden Regeln 12190 (TCP) und 35000 (UDP) um das Schlüsselwort `log`:

```
$ sudo ipfw list
...
12190 deny log tcp from any to any
...
35000 deny log udp from any to any in
...
```

Jeder erfolglose Zugriff auf den durch die Firewall abgesicherten Rechner resultiert auf diese Weise in einem Protokolleintrag. Das Protokoll kann wie üblich mit dem Dienstprogramm Konsole oder im Terminal überwacht werden:

```
$ tail -f /var/log/ipfw.log
...
Jul 23 20:37:10 Tiger-Server ipfw:  12190 Deny
TCP 192.168.1.42:49180 192.168.1.1:80 in via en1
Jul 23 20:37:13 Tiger-Server ipfw:  12190 Deny
TCP 192.168.1.42:49180 192.168.1.1:80 in via en1
Jul 23 20:37:16 Tiger-Server ipfw:  12190 Deny
TCP 192.168.1.42:49180 192.168.1.1:80 in via en1
Jul 23 20:37:19 Tiger-Server ipfw:  12190 Deny
TCP 192.168.1.42:49180 192.168.1.1:80 in via en1
...
```

Über die Option „Tarn-Modus aktivieren“ kann man hingegen die Antworten auf Pings unterdrücken. Der Befehl `ping` sendet ICMP-Pakete vom Typ *Echo Request* (Kennziffer 8) an den angegebenen Host. Der Host antwortet mit ICMP-Paketen vom Typ *Echo Reply* (Kennziffer 0). Anhand dieser Antworten lässt sich feststellen, ob ein Host erreichbar und eingeschaltet ist. Um den Host zu „verstecken“, kann man mit Hilfe der Firewall die entsprechenden Pakete blockieren.

Die Tarn-Option in der Systemeinstellung Sharing fügt dazu einfach eine weitere, auf ICMP-Pakete anwendbare Regel[12] ein:

```
$ sudo ipfw list
...
20000 deny log icmp from any to me in icmptypes 8
...
```

Um die Firewall wieder komplett zu öffnen, muss man in den Systemeinstellungen Sharing auf den Stop-Knopf im Bereich Firewall klicken. Dadurch werden alle Regeln (bis auf die Standardregel 65535) gelöscht. Die innerhalb der Systemeinstellung Sharing vorgenommenen Einstellungen merkt sich Mac OS X in der Voreinstellungsdatei `/Library/Preferences/com.apple.sharing.firewall.plist`, so dass diese nach einem erneuten Aktivieren der Firewall sofort wieder zur Verfügung stehen.

Statt über Systemeinstellungen Sharing kann man die Regelliste aber selbstverständlich auch direkt mit dem `ipfw`-Befehl manipulieren. Neue Regeln lassen sich mit dem Unterbefehl `add` hinzufügen, vorhandene einzeln mit dem Unterbefehl `delete` und komplett mit dem Unterbefehl `flush` löschen. Wie bereits in den Beispielen dargestellt, kann man mit dem Unterbefehl `list` die komplette Regelliste ausgeben:

```
$ sudo ipfw list
65535 allow ip from any to any
$ sudo ipfw add 10 allow tcp from any to any out
00010 allow tcp from any to any out
$ sudo ipfw add 20 allow tcp from any to any established
00020 allow tcp from any to any established
$ sudo ipfw add 65000 deny tcp from any to any
65000 deny tcp from any to any
```

[12] Anwendungen dieser Regel werden mitprotokolliert, sofern die Firewall-Protokollierung eingeschaltet wurde.

```
$ sudo ipfw list
00010 allow tcp from any to any out
00020 allow tcp from any to any established
65000 deny tcp from any to any
65535 allow ip from any to any
$ sudo ipfw delete 65000
$ sudo ipfw list
00010 allow tcp from any to any out
00020 allow tcp from any to any established
65535 allow ip from any to any
$ sudo ipfw flush
Are you sure? [yn] y

Flushed all rules.
```

Auf diese Weise könnte man z. B. den Zugang zu TCP-basierten Diensten auf einen bestimmten Host (oder ein bestimmtes Netzwerk) beschränken:

```
$ sudo ipfw flush
Are you sure? [yn] y

Flushed all rules.
$ sudo ipfw list
65535 allow ip from any to any
$ sudo ipfw add 1000 deny tcp from not 192.168.0.22 to me
  afpovertcp setup
01000 deny tcp from not 192.168.0.22 to me 548 setup
$ sudo ipfw list
01000 deny tcp from not 192.168.0.22 to me 548 setup
65535 allow ip from any to any
```

Im obigen Beispiel beschränken wir den TCP-Verbindungsaufbau (setup) für Personal File Sharing (afpovertcp bzw. Port 548) auf den Host mit der IP-Adresse 192.168.0.22.

Um Konflikte zu vermeiden, sollte man sich jedoch auf eine einzige Eingabemethode für Regeln beschränken und insbesondere die Firewall nicht über Systemeinstellungen Sharing aktivieren, wenn man bereits direkt über `ipfw` einige Regeln eingegeben hat. Glücklicherweise erkennt die GUI in den meisten Fällen selbst die Veränderungen und deaktiviert die Firewall-Einstellungen (Abb. 3.26). Die Firewall-Einstellungen lassen sich nach einer solchen Deaktivierung erst nach einer vollständigen Löschung (mittels `ipfw flush`) der direkt eingegebenen Regeln wieder aktivieren.

Im Gegensatz zu den über Systemeinstellungen Sharing definierten Regeln sind direkt eingegebene Regeln lediglich bis zum nächsten Neustart gültig[13]. Um eigene Regeln nach einem Neustart automatisch zu installieren und nicht manuell neu eingeben zu müssen, sollte ein eigenes StartupItem, z. B. /Library/StartupItems/Firewall, mit den entsprechenden Aufrufen von `ipfw` erstellt werden. Um die Ausgaben dieses Befehls in diesem Fall zu unterdrücken, sollte man im StartupItem-Skript die Option `-q` verwenden:

[13] Die über die GUI erstellten Regeln werden in /Library/Preferences/com.apple.sharing.firewall.plist gespeichert und nach einem Neustart automatisch wiederhergestellt.

Abb. 3.26. Um Konflikte zu vermeiden, sollte man sich auf eine Eingabemethode für Regeln beschränken. Immerhin bekommt man (leider nicht immer) eine (etwas irreführende) Fehlermeldung.

```
#!/bin/sh
/sbin/ipfw -q flush
/sbin/ipfw -q add 1000 deny tcp from not 192.168.0.22 to me
              afpovertcp setup
...
```

3.4.3 NAT

Die automatische Adressübersetzung (engl. Network Address Translation, kurz NAT) ist heutzutage weit verbreitet. Da nur verhältnismäßig wenige Organisationen über einen größeren Pool von öffentlichen IP-Adressen verfügen, ist es gängige Praxis, dass selbst große Netzwerke mit Hilfe von privaten IP-Adressen eingerichtet werden. NAT ist dabei ein häufig gegangener Weg, um

Hosts in solchen privaten Netzwerken die Kommunikation mit dem Rest der Welt zu ermöglichen.

In der Praxis setzt man für NAT häufig kompakte Router ein, die neben NAT auch gleich das Routing zwischen dem lokalen Netz und dem jeweiligen Zugangsanbieter übernehmen. Häufig bieten diese Geräte auch weitere Funktionen an, wie z. B. einen rudimentären DHCP-Server. Im Abschnitt 3.4.1 haben wir im Kontext von DHCP die Funktionen solcher Router kurz angerissen.

Sehr häufig ist ein solcher dedizierter NAT-Router die mit Abstand einfachste und wirtschaftlichste Lösung. Selbstverständlich beherrscht aber auch Mac OS X NAT. Damit kann ein Mac-OS-X-Rechner die Funktion eines Routers/Gateways übernehmen.

Am einfachsten kann die NAT-Funktion von Mac OS X über die Systemeinstellung Sharing aktiviert werden. Über die Schaltfläche *Internet* gelangt man hier zu einem Dialog, der die Aktivierung der NAT-Funktion erlaubt (Abb. 3.27).

Über das Pop-up-Menü „Verbindung gemeinsam nutzen“ ist die *externe* Netzwerkschnittstelle auszuwählen, also die Schnittstelle, über die der Rest der Welt erreicht werden kann, und die mit einer öffentlichen IP-Adresse konfiguriert ist. Aus der Liste „Mit Computern, die Folgendes verwenden“ ist anschließend die *interne* Netzwerkschnittstelle auszuwählen. Dabei sollte man darauf achten, dass der externen Netzwerkschnittstelle eine höhere Priorität zugewiesen wird als der internen, damit das Routing funktioniert und NAT tatsächlich aktiviert wird.

Durch den Start von *Internet Sharing* löst man dann jede Menge Funktionen aus. Zum einen wird die Routingfunktion im Betriebssystemkern aktiviert, was man mit dem `syctl`-Befehl leicht selbst überprüfen kann:

```
$ sysctl net.inet.ip.forwarding
net.inet.ip.forwarding: 1
```

Ist die Routingfunktion aktiv, erscheint als Ausgabe `net.inet.ip.forwarding: 1`, ansonsten erscheint `net.inet.ip.forwarding: 0`.

Zum anderen wird `bootpd` gestartet[14] und in der NetInfo-Datenbank als DHCP-Server für die interne Netzwerkschnittstelle konfiguriert:

```
$ ps ax -o command | grep bootpd | grep -v grep
/usr/libexec/bootpd -P

$ nicl / -read /config/dhcp
name: dhcp
bootp_enabled:
dhcp_enabled: en1
reply_threshold_seconds: 4
detect_other_dhcp_server: 1
```

[14] Die Option -P ist in der mit Mac OS X 10.4 mitgelieferten man-Page zu bootpd leider nicht dokumentiert.

Abb. 3.27. Die einfachste Möglichkeit, die NAT-Funktion von Mac OS X zu nutzen bietet sich über die Verwendungen der Systemeinstellungen Sharing.

Im Beispiel haben wir die sekundäre Ethernet-Schnittstelle en1 als interne Schnittstelle gewählt. In der NetInfo-Datenbank findet sich deshalb das Attribut `dhcp_enabled` mit dem Wert `en1`. Das Attribut `bootp_enabled`, hier ohne einen Wert, verhindert, dass neben DHCP auch der BOOTP-Server aktiviert wird.

Als IP-Adressenpool wird ein scheinbar relativ willkürlich gewähltes privates Netzwerk mit 253 freien Hostadressen verwendet:

```
$ nicl / -list /config/dhcp/subnets
95          192.168.2

$ nicl / -read /config/dhcp/subnets/192.168.2
name: 192.168.2
net_address: 192.168.2.0
net_mask: 255.255.255.0
dhcp_router: 192.168.2.1
```

```
lease_max: 3600
client_types: dhcp
dhcp_domain_name_server: 192.168.2.1
net_range: 192.168.2.2 192.168.2.254
_creator: com.apple.nat
```

Der eigenen internen Netzwerkschnittstelle wird dabei die erste Adresse aus dem Pool zugewiesen – z. B. 192.168.2.1. Diese Zuweisung wird zwar im Konfigurationsbereich von Systemeinstellungen Netzwerk nicht angezeigt, kann aber mit Hilfe des Befehls `ifconfig` überprüft werden:

```
$ ifconfig en1 inet
en1: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>
     mtu 1500
        inet 192.168.2.1 netmask 0xffffff00 broadcast 192.168.2.255
```

Abb. 3.28. Die externe Netzwerkschnittstelle (hier „Ethernet (integriert)“) muss eine höhere Priorität haben als die interne (hier „PCI-Ethernet-Karte 3“).

Die eigentliche NAT-Funktion wird dann mit Hilfe von zwei weiteren Komponenten realisiert: `natd` und der Firewall. Der eigentliche NAT-Daemon `natd` verändert IP-Pakete, indem er die IP-Adresse und die Port-Nummer des Absenders dem NAT-Verfahren entsprechend modifiziert. Die Firewall leitet alle zwischen dem internen und dem externen Netzwerk ausgetauschten Pakete über den NAT-Daemon.

Der NAT-Daemon wird von Internet Sharing mit den folgenden Optionen aktiviert:

```
$ ps axwww -o command | grep natd | grep -v grep
/usr/sbin/natd -alias_address 192.168.0.10 -interface en0
               -use_sockets -same_ports -unregistered_only
               -dynamic -clamp_mss -enable_natportmap
               -natportmap_interface en1
```

Mit der Option `-alias_address` teilt Internet Sharing `natd` die öffentliche IP-Adresse der vom Benutzer ausgewählten externen Netzwerkschnittstelle mit – hier 192.168.0.10. Genau diese Alias-Adresse wird `natd` bei allen ausgehenden Paketen als Absenderadresse eingetragen.

Als nächstes wird mit Hilfe der Option `-interface` die Bezeichnung der externen Netzwerkschnittstelle übergeben – hier en0. Diese Angabe ist eigentlich nur erforderlich, wenn keine Alias-Adresse angegeben wurde, und daher an dieser Stelle redundant.

Die weiteren Optionen sind im Prinzip nur Optimierungen:

- die Option `-use_sockets` erhöht die Kompatibilität mit FTP und IRC DCC durch die Verwendung eines dedizierten Kommunikationsendpunkts (Socket) für diese Protokolle.
- Über die Option `-same_ports` wird festgelegt, dass `natd` nach Möglichkeit nur die IP-Adresse und nicht die Port-Nummer des Absenders modifizieren soll. Das erhöht die Kompatibilität mit bestimmten Protokollen, wie z. B. RPC, ist aber nur möglich, solange nicht zwei (oder mehr) private Hosts die gleichen Port-Nummern verwenden. Ist letzteres der Fall, wird `natd` trotz dieser Option die Port-Nummern modifizieren.
- Die Option `-unregistered_only` lässt `natd` nur Pakete von Absendern mit einer privaten IP-Adresse verändern. Sind über das interne Netzwerk nur Hosts mit privaten IP-Adressen zu erreichen, ist diese Option überflüssig.
- Die Option `-dynamic` ergänzt die Option `-interface`. Mit ihr wird `natd` angewiesen, auf Änderungen der öffentlichen IP-Adresse zu achten und die verwendete Alias-Adresse gegebenenfalls entsprechend automatisch zu ändern.
- Die Option `-clamp_mss` veranlasst `natd` dazu, die maximale Größe von TCP-Segmenten (engl. Maximum Segment Size, kurz MSS) zu verändern. Das ist regelmäßig dann notwendig, wenn das interne und das externe Netzwerk unterschiedliche MTUs aufweisen. Ein typisches Beispiel wäre

Ethernet als internes Netzwerk mit einer MTU von 1500 Bytes und PPPoE als externes Netzwerk mit einer MTU von 1492 Bytes. Ohne diese Option kann es in einem solchen Fall zu Kommunikationsproblemen mit Hosts in bestimmten Netzwerken kommen.

Die beiden Optionen `enable_natportmap` und `natportmap_interface` sind z. Z. nicht dokumentiert.

Internet Sharing leitet alle über die externe Netzwerkschnittstelle ankommenden und ausgehenden Pakete über `natd` mit Hilfe einer entsprechenden Firewall-Regel. Solange der Benutzer die Firewall-Funktionalität in Systemeinstellungen Sharing nicht aktiviert hat oder direkt selbst eigene Regeln definiert hat, sieht die Konfiguration von `ipfw` nach der Aktivierung von Internet Sharing wie folgt aus:

```
$ sudo ipfw list
00010 divert 8668 ip from any to any via en0
65535 allow ip from any to any
```

Die Regel Nummer 10 hat die höchste Priorität und leitet alle über die (externe) Netzwerkschnittstelle en0 ankommenden und ausgehenden IP-Pakete auf den von `natd` verwendeten Port 8668.

```
$ grep 8668 /etc/services
natd            8668/divert # Network Address Translation

$ sudo lsof -i:8668
COMMAND PID USER   FD   TYPE     DEVICE SIZE/OFF   NODE NAME
natd    296 root    4u  IPv4 0x0328aee0      0t0 DIVERT *:8668
```

Nachdem das Paket von `natd` bearbeitet worden ist, wird es an die Firewall-Regel mit der nächsthöheren Nummer weitergegeben – im obigen Fall die Standardregel 65535, die es einfach weiter passieren lässt.

Natürlich kann man die NAT-Funktion auch direkt aktivieren, ohne auf Internet Sharing zurückzugreifen. Das ist insbesondere dann interessant, wenn man eine besondere DHCP-Konfiguration benötigt oder weitere Optionen von `natd` verwenden möchte.

Besonders häufig benötigt man in der Praxis eine feste Port-Weiterleitung. Mit Hilfe der Port-Weiterleitung lassen sich an die IP-Adresse der externen Schnittstelle gerichtete TCP- und UDP-Verbindungen gezielt auf Rechner im privaten, internen Netzwerk umleiten.

In den obigen Beispielen hatte die externe Netzwerkschnittstelle die IP-Adresse 192.168.0.10. Falls wir nun vorhätten, einen Webserver statt auf dem NAT-Host auf einem anderen Host im privaten Netzwerk dahinter zu betreiben, z. B. auf dem Host 192.168.2.100, müssten wir eine feste Port-Weiterleitung aktivieren.

Dazu müssten wir `natd` mitteilen, dass alle an Host 192.168.0.10 und den Port 80 gerichteten TCP-Segmente an den Host 192.168.2.100 und dessen Port 80 umzuleiten sind[15]. Eine Port-Weiterleitung entspricht damit also

[15] Die Port-Nummern können – wie in diesem Fall – übereinstimmen, müssen aber nicht.

einfach einem permanenten Eintrag in der von `natd` verwalteten Adressübersetzungstabelle.

Ein solcher Eintrag lässt sich mit Hilfe der Option `-redirect_port` erzeugen, z. B. indem `natd` wie folgt aufruft:

```
$ sudo natd -alias_address 192.168.0.19 -interface en1
            -use_sockets -same_ports -unregistered_only
            -dynamic -clamp_mss -redirect_port
             tcp 192.168.2.100:80 80
```

Noch besser ist es in diesem Fall aber, mit Hilfe der Option `-f` eine Konfigurationsdatei zu verwenden, um die nun doch recht lange Liste von Parametern dauerhaft abzuspeichern. In der Konfigurationsdatei darf pro Zeile jeweils nur ein Parameter angegeben werden. Parameter, die in der Kommandozeile normalerweise ohne Argumente angegeben werden, können in der Konfigurationsdatei mit dem Argument `yes` aktiviert, und mit dem Argument `no` deaktiviert werden. Den obigen Aufruf könnte man also zu `natd -f natd.conf` verkürzen, indem man eine Datei `natd.conf` mit dem folgenden Inhalt erzeugt:

```
#
# natd Konfigurationsbeispiel
#

# Standardoptionen
alias_address 192.168.0.10
interface en0
use_sockets yes
same_ports yes
unregistered_only yes
clamp_mss yes

# Port-Weiterleitung
redirect_port tcp 192.168.2.100:80 80
```

Die mit `#` beginnenden Zeilen werden als Kommentare ignoriert. Leere Zeilen dürfen zur besseren Lesbarkeit eingefügt werden.

Um NAT bei jedem Systemstart automatisch zu aktivieren, kann man ein StartupItem anlegen, indem man `natd` startet und die Firewall konfiguriert. Das StartupItem-Skript könnte dann lauten:

```
#!/bin/sh

#
# /Library/StartupItems/NAT/NAT
#

# Starte natd
/usr/sbin/natd -f /Library/StartupItems/NAT/natd.conf

# Konfiguriere Firewall
/sbin/ipfw -f flush
/sbin/ipfw ipfw add 10 divert natd ip from any to any via en0
```

Der Speicherort der Konfigurationsdatei (hier `/Library/StartupItems/NAT/natd.conf`), deren Inhalt sowie die externe Netzwerkschnittstelle (hier en0) müssen natürlich an die eigenen Bedürfnisse angepasst werden.

Zusätzlich muss man außerdem noch das Routing aktivieren und der internen Netzwerkschnittstelle eine IP-Adresse zuweisen. Das Erstere lässt sich, wie bereits beschrieben, sehr einfach durch die Änderung des Eintrags `IPFORWARDING=-NO-` in `IPFORWARDING=-YES-` in der Datei `/etc/hostconfig` dauerhaft bewerkstelligen. Das Letztere erledigt man am einfachsten in den Systemeinstellungen Netzwerk.

In den meisten Fällen wird man neben NAT gleichzeitig auch DHCP aktivieren wollen, um im privaten Netzwerk die TCP/IP-Konfigurationsparameter automatisch zu verteilen. DHCP ist aber mitnichten eine Voraussetzung für NAT, da man die Hosts im privaten Netzwerk genauso gut auch manuell konfigurieren kann. Das ist natürlich mühsam, so dass man in den meisten Fällen auf einen DHCP-Server nicht verzichten wird. Der DHCP-Server kann dabei auf dem gleichen Host wie NAT laufen. Er kann aber ebenso auf einem separaten Rechner installiert werden.

3.4.4 DNS

Ein DNS-Server hat im Normalfall zwei voneinander völlig unabhängige Aufgaben zu übernehmen. Die eine Aufgabe besteht darin, den verhältnismäßig einfachen Resolvern der Hosts bei der Auflösung von DNS-Namen bzw. IP-Adressen zu helfen. Die andere, etwas komplexere Aufgabe besteht darin, für einen definierten Teil des DNS-Namensraums die Zuordnung von DNS-Namen zu IP-Adressen zu definieren und diesbezügliche Anfragen von Resolvern und anderen DNS-Servern zu beantworten.

Für die erste Aufgabe ist ein eigener DNS-Server normalerweise nicht erforderlich. Die meisten Internet-Zugangsanbieter stellen ihren Kunden einen (oder mehrere) DNS-Server zur Mitbenutzung zu diesem Zweck zur Verfügung. Ein eigener Server kann allerdings aufgrund von Caching eine Optimierung darstellen, wenn die DNS-Server des Zugangsanbieters nur über eine im Verhältnis zum lokalen Netzwerk sehr langsame Leitung zu erreichen sind.

Für die zweite Aufgabe kann man in privaten Netzwerken manchmal um die Einrichtung eines eigenen DNS-Servers herumkommen und sich mit einfacheren Alternativlösungen behelfen. Dazu gehören neben dem völligen Verzicht auf Hostnamen (was wirklich nur in sehr kleinen Netzwerken praktikabel ist) z. B. tabellengestützte Verfahren auf der Basis von Einträgen in `/etc/hosts/` oder im `/machines`-Verzeichnis der NetInfo-Datenbank. Ebenso kann man in kleineren Netzwerken oft ganz gut mit Bonjour (bzw. Multicast DNS, einem Teilaspekt von Rendevous) arbeiten, wenn alle verwendeten Hosts dies auch unterstützen.

In größeren und/oder heterogenen privaten Netzwerken stellt ein eigener DNS-Server jedoch eine große Vereinfachung dieser Aufgabe dar. Ohnehin

setzen manche für größere Netzwerke interessanten Dienste eine funktionierende DNS-Infrastruktur voraus: zu solchen Diensten gehören beispielsweise Kerberos und NetBoot.

In öffentlichen Netzwerken, d. h. TCP/IP-Netzwerken mit öffentlichen IP-Adressen, kommt man um einen DNS-Server nicht herum, wobei man in manchen Fällen auch hier auf den DNS-Dienst des Zugangsanbieters zurückgreifen kann[16].

Die für den Betrieb eines DNS-Servers notwendige Software, BIND, wird mit Mac OS X mitgeliefert. BIND ist eine Abkürzung für *Berkeley Internet Name Domain* und ist die heutzutage am weitesten verbreitete Implementierung von DNS. BIND besteht im Wesentlichen aus zwei Komponenten: dem DNS-Server named und dem DNS-Resolver. Mac OS X 10.4 enthält die Version 9.2.2 von BIND:

```
$ named -v
BIND 9.2.2
```

BIND ist sehr mächtig und kann abhängig von den individuellen Bedürfnissen auf sehr vielfältige Weise konfiguriert werden. Das Spektrum reicht von einem einfachen DNS-Cache ohne jegliche Zuständigkeiten für irgendwelche Teile des DNS-Namensraums, bis zu sehr komplexen Konfigurationen, welche auch die ausgefallensten Wünsche von Internet-Zugangsanbietern und von Betreibern der Internet-Infrastruktur abdecken müssen.

Ein einfacher DNS-Cache

Der mit Abstand einfachste Fall einer BIND-Konfiguration ist der Betrieb als reiner DNS-Cache. Für diese Betriebsvariante ist BIND unter Mac OS X bereits vorkonfiguriert. Um unter Mac OS X 10.3 den DNS-Server automatisch beim Systemstart zu aktivieren, muss in der Datei `/etc/hostconfig` der Eintrag `DNSSERVER=-NO-` in `DNSSERVER=-YES-` geändert werden.

Beim nächsten Systemstart sorgt dann das StartupItem BIND (`/System/Library/StartupItems/BIND`) dafür, dass der DNS-Server gestartet wird:

```
...
StartService ()
{
    if [ "${DNSSERVER:=-NO-}" = "-YES-" ]; then
        ConsoleMessage "Starting named"
        named
    fi
}
...
```

[16] Das ist regelmäßig z. B. dann der Fall, wenn man nur einen öffentlichen Rechner im Rechenzentrum des Zugangsanbieters betreibt, oder gar nur Plattenplatz auf einem öffentlichen Rechner des Zugangsanbieters gemietet hat, um dort seine Website zu veröffentlichen.

Unter Mac OS X 10.4 wird `named` als Launch Daemon gestartet. Dazu existiert bereits, ähnlich wie unter Mac OS X 10.3, eine entsprechende Konfigurationsdatei:

```
$ cat /System/Library/LaunchDaemons/org.isc.named.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Disabled</key>
        <true/>
        <key>Label</key>
        <string>org.isc.named</string>
        <key>OnDemand</key>
        <false/>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/named</string>
                <string>-f</string>
        </array>
        <key>ServiceIPC</key>
        <false/>
</dict>
</plist>
```

Im Prinzip könnte man damit `named` einfach mit `launchctl` starten:

```
$ sudo launchctl load -w \
> /System/Library/LaunchDaemons/org.isc.named.plist
```

In der Praxis fehlt in Mac OS X 10.4 jedoch die Datei `/etc/rndc.key`, so dass `named` nicht gestartet werden kann. Stattdessen erscheint eine entsprechende Fehlermeldung auf der Konsole:

```
$ tail -f /Library/Logs/Console/501/console.log
...
Jul 24 14:21:50 Tiger-Server named[400]:
/private/etc/named.conf:4: open: /etc/rndc.key:
file not found
Jul 24 14:21:50 Tiger-Server named[400]:
loading configuration: file not found
Jul 24 14:21:50 Tiger-Server named[400]:
exiting (due to fatal error)
```

Dieser Fehler lässt sich entweder durch Abänderung der Konfigurationsdatei `/etc/named.conf` oder durch das Erstellen der fehlenden Datei beheben. Letzteres lässt sich leicht mit dem Befehl `rndc-confgen` bewerkstelligen:

```
$ sudo rndc-confgen -a
```

Anschließend kann man `named` wie oben beschrieben starten. Da `named` im Gegensatz zu den meisten anderen Launch Daemons sofort gestartet wird (das Schlüsselwort `OnDemand` hat den Wert `false`), kann man sofort sehen, ob der Prozess gestartet wurde oder nicht:

```
$ sudo launchctl load -w \
> /System/Library/LaunchDaemons/org.isc.named.plist

$ ps ax -o command | grep named | grep -v grep
/usr/sbin/named -f
```

Sobald der DNS-Server als DNS-Cache gestartet wurde, kann man ihn sofort wie jeden anderen DNS-Server benutzen. Auf der Client-Seite genügt es, die IP-Adresse des Rechners, auf dem der DNS-Cache läuft, als DNS-Server einzutragen (über Systemeinstellungen Netzwerk oder über eine entsprechende Konfiguration des DHCP-Servers). Damit teilt man dem DNS-Resolver des Clients mit, alle DNS-Anfragen an den DNS-Cache weiterzuleiten.

Um dann z. B. das erste Mal einen DNS-Namen wie z. B. *www.apple.com* aufzulösen, sendet der DNS-Resolver des Clients eine entsprechende Anfrage and den DNS-Cache. Da unmittelbar nach dem Start der DNS-Cache weder die IP-Adresse von *www.apple.com*, noch die für die *apple.com*-Domain oder die *com* Top-Level-Domain (TLD) zuständigen DNS-Server kennt, bleibt ihm nichts anderes übrig, als einen der 13 Root-Server zu kontaktieren und dort nach der IP-Adresse von *www.apple.com* zu fragen.

Der Root-Server wird daraufhin aber lediglich mit den Adressen der ebenfalls 13 für die *com*-Domain zuständigen DNS-Server antworten, woraufhin unser DNS-Cache einen davon kontaktieren muss, um erneut nach der IP-Adresse von *www.apple.com* zu fragen. Ein für die *com* TLD zuständiger DNS-Server kennt selbst aber auch nur die Adressen der sechs DNS-Server, an die die Verantwortung für die *apple.com*-Domain delegiert wurde. Damit ist also mindestens noch eine weitere Anfrage unseres DNS-Caches erforderlich, um einen dieser unter der Verantwortung des Domaininhabers (in dem Fall Apple Computer) stehenden DNS-Server zu kontaktieren.

Im Normalfall wäre hier die Suche zu Ende und ein Apple-DNS-Server würde die IP-Adresse von *www.apple.com* auf Anfrage offenbaren. In diesem speziellen Fall wiederholt sich das Verfahren noch einige Male, da es sich bei *www.apple.com* lediglich um ein Alias handelt, welches auf den Namen *www.apple.com.akadns.net* des Dienstleisters Akamai zeigt.

Das macht eine erneute Anfrage an einen Root-Server erforderlich, um die Adressen der 13 für die *net* TLD zuständigen DNS-Server zu erfahren. Einen von diesen kann der DNS-Cache dann wiederum nach den für die Akamai-Domain *akadns.net* zuständigen DNS-Servern fragen. Erst von einem dieser elf Server kann der DNS-Cache dann endlich die IP-Adresse von *www.apple.com.akadns.net* bekommen und an den DNS-Resolver des Clients weiterleiten.

Um diese vielen Anfragen nicht wiederholen zu müssen, speichert der DNS-Cache alle Ergebnisse und kann nachfolgende Anfragen entsprechend schneller direkt selbst beantworten. Dabei werden auch sämtliche Teilergebnisse berücksichtigt. Eine Anfrage nach *www.mac.com* erfordert dann keine Kontaktaufnahme mit den Root-Servern mehr, da die für die *com* TLD zuständigen Server bereits bekannt sind.

Je länger der DNS-Cache läuft, desto mehr Anfragen kann er direkt selbst beantworten. Natürlich muss von Zeit zu Zeit geprüft werden, ob die gesammelten Daten noch aktuell sind. Jede in einer Antwort erhaltene Angabe ist dazu mit einer vom Administrator der jeweiligen Domain festgelegten Zeitspanne versehen, die über ihre Gültigkeitsdauer entscheidet. Wird diese TTL (engl. time to live) genannte Zeitspanne überschritten, muss der DNS-Cache erneut nachfragen.

In der Regel sind für jede Domain mindestens zwei DNS-Server zuständig (für die Root-Domain und die TLDs sogar 13). Der DNS-Cache probiert bei wiederholten Anfragen an die gleiche Domain alle DNS-Server einer Domain nacheinander in einer zufälligen Reihenfolge aus, um denjenigen zu ermitteln, der am schnellsten antwortet. Bei weiteren Anfragen wird dann der DNS-Server mit den besten Antwortzeiten verwendet.

Damit das alles funktioniert, muss der DNS-Server `named` zumindest die IP-Adressen der Root-Server kennen. Die Angabe, wo diese zu finden sind, findet sich nebst einigen weiteren Konfigurationsparametern in der Datei `/etc/named.conf`. Aus dieser Datei werden, sofern beim Aufruf nichts anderes angegeben wurde, beim Start des DNS-Servers alle Konfigurationsparameter gelesen. Im Auslieferungszustand von Mac OS X hat diese Datei folgendes Aussehen:

```
$ cat /etc/named.conf
//
// Include keys file
//
include "/etc/rndc.key";

// Declares control channels to be used by the rndc utility.
//
// It is recommended that 127.0.0.1 be the only address used.
// This also allows non-privileged users on the local host to manage
// your name server.

//
// Default controls
//
controls {
        inet 127.0.0.1 port 54 allow {any;}
        keys { "rndc-key"; };
};

options {
        directory "/var/named";
        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
        // query-source address * port 53;
};
//
```

```
// a caching only nameserver config
//
zone "." IN {
        type hint;
        file "named.ca";
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

logging {
        category default {
                _default_log;
        };

        channel _default_log  {
                file "/Library/Logs/named.log";
                severity info;
                print-time yes;
        };
};
Tiger-Server:~ admin
```

Die Zeichenkombination `//` oder das Zeichen `#` leiten einen Kommentar ein, der mit dem Ende der jeweiligen Zeile endet. Kommentare, die sich über mehrere Zeilen erstrecken, kann man durch die Zeichenkombinationen `/*` und `*/` einklammern.

Die oben dargestellte, von Apple ausgelieferte BIND-Konfigurationsdatei lässt sich grob in vier Teilbereiche gliedern. Der erste Bereich besteht dabei aus der Anweisung `controls`, der zweite Bereich aus der Anweisung `options`, der dritte Bereich aus drei Anweisungen vom Typ `zone` und der vierte Bereich aus der Anweisung `logging`. Dabei definieren die `zone`-Anweisungen die Zuständigkeit des DNS-Servers für Teile des DNS-Namensraums, die DNS-Zonen genannt werden.

Der DNS-Server kann während der Laufzeit mit Hilfe des Befehls `rndc` gesteuert werden. Die Anweisung `controls` legt hier fest, dass der Zugriff nur lokal (und nicht über das Netzwerk) und nur über den Port 54 erlaubt ist.

Über die Anweisung `options` kann das Verhalten des DNS-Servers in vielfältigster Weise beeinflusst werden. Für die hier angestrebte einfache Konfiguration als DNS-Cache wird lediglich über die Option `directory` das Arbeitsverzeichnis für alle folgenden Anweisungen festgelegt. Die Anweisung spart damit etwas Tipparbeit weiter unten, da bei Dateiangaben lediglich der Dateiname und nicht der komplette Pfad angegeben werden muss.

Die erste `zone`-Anweisung ist ein (wichtiger) Spezialfall und teilt dem DNS-Server mit, dass die IP-Adressen der Root-Server der Datei `/var/named/named.ca` zu entnehmen sind. Da diese IP-Adressen manchmal Änderungen unterliegen, kann man die jeweils aktuellste Liste vom Rechner `ftp.internic. net` über FTP beziehen. Die Datei ist dort im Verzeichnis `domain` unter der Bezeichnung `named.root` abgelegt. Einer der Root-Server wird beim Start allerdings ohnehin kontaktiert um die jeweils aktuellste Liste zu bekommen. Falls diese Anweisung fehlt, wird eine innerhalb des DNS-Servers fest vorgegebene (alte) Liste von DNS-Servern verwendet.

Die zweite `zone`-Anweisung definiert die Zuständigkeit des Caches für den DNS-Namen `localhost`. Das Ziel ist hier, DNS-Anfragen nach diesem Namen direkt mit der IP-Adresse 127.0.0.1 zu beantworten. Durch das Attribut `type master` wird festgelegt, dass dieser DNS-Server selbst lokal die hierfür erforderlichen Informationen besitzt und nicht etwa von einem anderen Server regelmäßig (`type slave`) oder gar nur bei Bedarf (`type forward`), bezieht. Das Attribut `file` legt fest, dass diese Informationen der Datei `/var/named/localhost.zone` zu entnehmen sind.

Die dritte `zone`-Anweisung ist schließlich das direkte Spiegelbild der zweiten. Hier wird die Zuständigkeit des Caches für IP-Adressen aus dem Localhost-Bereich 127.0.0.0/24 definiert. Analog sollen hier Anfragen nach der IP-Adresse 127.0.0.1 direkt mit dem Namen `localhost` beantwortet werden. Auch hier wird über die Attribute `type` und `file` festgelegt, dass die dazu nötigen Informationen der Datei `/var/named/named.local` zu entnehmen sind.

Fernsteuerung mit rndc

Der DNS-Server `named` lässt sich nur sehr grob über Signale steuern. Mit Hilfe eines SIGHUP-Signals kann die komplette Konfiguration (z. B. nach Änderungen) neu eingelesen werden, und mit Hilfe des SIGINT- oder eines SIGTERM-Signals kann der DNS-Server „sanft“ beendet werden. Das ist alles.

Deutlich erweiterte Steuerungsmöglichkeiten bietet der Befehl `rndc`. Man kann mit diesem Befehl über die Optionen `reload`, `refresh` und `reconfig` nicht nur sehr präzise angeben, welche Konfigurationsdaten neu eingelesen werden sollen, sondern auch die Protokollierung einschalten (`querylog`), den aktuellen Status ausgeben (`status`) oder sogar den Inhalt des DNS-Caches ausgeben lassen (`dumpdb`):

```
$ rndc
Usage: rndc [-c config] [-s server] [-p port]
        [-k key-file ] [-y key] [-V] command

command is one of the following:

  reload        Reload configuration file and zones.
  reload zone [class [view]]
```

```
                Reload a single zone.
  refresh zone [class [view]]
                Schedule immediate maintenance for a zone.
  reconfig      Reload configuration file and new zones only.
  stats         Write server statistics to the statistics file.
  querylog      Toggle query logging.
  dumpdb        Dump cache(s) to the dump file (named_dump.db).
  stop          Save pending updates to master files and stop
                the server.
  halt          Stop the server without saving pending updates.
  trace         Increment debugging level by one.
  trace level   Change the debugging level.
  notrace       Set debugging level to 0.
  flush         Flushes all of the server's caches.
  flush [view]  Flushes the server's cache for a view.
  status        Display status of the server.
  *restart      Restart the server.

* == not yet implemented
Version: 9.2.2
```

Obwohl die mit Mac OS X 10.4 ausgelieferte BIND-Konfigurationsdatei explizit die Anweisung `controls` enthält, funktioniert der Aufruf von `rndc` allerdings nicht auf Anhieb:

```
$ sudo named
$ sudo rndc status
rndc: neither /private/etc/rndc.conf  nor  /private/etc/rndc.
key was found
```

Ein Grund dafür ist das fehlende Passwort für die Kommunikation zwischen `named` und `rndc`. Am einfachsten kann dieses Passwort mit dem Befehl `rndc-confgen` gesetzt werden. Beim Aufruf mit der Option `-a` erzeugt dieser Befehl ein Passwort in der (nur für Root lesbaren) Datei `/etc/rndc.key`, die automatisch sowohl von `named` als auch von `rndc` gelesen wird. Das alleine genügt jedoch nicht:

```
$ sudo rndc-confgen -a
$ sudo killall -HUP named
$ sudo rndc status
rndc: connect failed: connection refused
```

Der zweite Stolperstein ist die explizite Erwähnung des Kommunikationsports 54 in der `controls`-Anweisung innerhalb der Datei `/etc/named.conf`. Damit erwartet `named` Steuerungsverbindungen ausschließlich auf dem Port 54. Da `rndc` normalerweise den Port 953 verwendet, muss man entweder die Datei `/etc/named.conf` entsprechend editieren oder beim Aufruf von `rndc` die Option `-p` verwenden:

```
$ sudo rndc -p 54 status
number of zones: 4
debug level: 0
xfers running: 0
xfers deferred: 0
soa queries in progress: 0
```

```
query logging is OFF
server is up and running
```

Beachtet man diese zwei Besonderheiten, steht der Verwendung von `rndc` nichts mehr im Wege.

Da man mit der Option `flush` den ganzen DNS-Cache leeren und mit der Option `dumpdb` seinen Inhalt in die Datei `/var/named/named_dump.db` ausgeben kann, kann man sich mit Hilfe von `rndc` sehr gut noch mal die Arbeitsweise von DNS vergegenwärtigen. Im folgenden Beispiel wird in Folge der DNS-Cache geleert, um anschließend mit Hilfe eines einzelnen Pings (der den DNS-Namen *www.apple.com* auflöst) gefüllt und schließlich ausgegeben zu werden:

```
$ sudo rndc -p 54 flush

$ ping -c 1 www.apple.com
PING www.apple.com.akadns.net (17.254.0.91): 56 data bytes
64 bytes from 17.254.0.91: icmp_seq=0 ttl=53 time=213.79 ms

--- www.apple.com.akadns.net ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 213.79/213.79/213.79 ms

$ sudo rndc -p54 dumpdb
```

Die Datei `/var/named/named_dump.db` enthält anschließend alle für die Auflösung des DNS-Namens *www.apple.com* nötigen Zwischenergebnisse.

```
$ cat /var/named/named_dump.db
;
; Cache dump of view '_default'
;
$DATE 20040726185729
; authanswer
.                       518364  IN NS   A.ROOT-SERVERS.NET.
                        518364  IN NS   B.ROOT-SERVERS.NET.
                        518364  IN NS   C.ROOT-SERVERS.NET.
                        518364  IN NS   D.ROOT-SERVERS.NET.
                        518364  IN NS   E.ROOT-SERVERS.NET.
                        518364  IN NS   F.ROOT-SERVERS.NET.
                        518364  IN NS   G.ROOT-SERVERS.NET.
                        518364  IN NS   H.ROOT-SERVERS.NET.
                        518364  IN NS   I.ROOT-SERVERS.NET.
                        518364  IN NS   J.ROOT-SERVERS.NET.
                        518364  IN NS   K.ROOT-SERVERS.NET.
                        518364  IN NS   L.ROOT-SERVERS.NET.
                        518364  IN NS   M.ROOT-SERVERS.NET.
; glue
com.                    172762  NS      A.GTLD-SERVERS.NET.
                        172762  NS      B.GTLD-SERVERS.NET.
                        172762  NS      C.GTLD-SERVERS.NET.
                        172762  NS      D.GTLD-SERVERS.NET.
                        172762  NS      E.GTLD-SERVERS.NET.
                        172762  NS      F.GTLD-SERVERS.NET.
                        172762  NS      G.GTLD-SERVERS.NET.
                        172762  NS      H.GTLD-SERVERS.NET.
                        172762  NS      I.GTLD-SERVERS.NET.
```

```
                        172762  NS      J.GTLD-SERVERS.NET.
                        172762  NS      K.GTLD-SERVERS.NET.
                        172762  NS      L.GTLD-SERVERS.NET.
                        172762  NS      M.GTLD-SERVERS.NET.
; glue
apple.com.              172762  NS      nserver.asia.apple.com.
                        172762  NS      nserver.euro.apple.com.
                        172762  NS      nserver.apple.com.
                        172762  NS      nserver2.apple.com.
                        172762  NS      nserver3.apple.com.
                        172762  NS      nserver4.apple.com.
; glue
nserver.asia.apple.com. 172762  A       203.120.14.5
; glue
nserver.euro.apple.com. 172762  A       17.72.133.64
; glue
nserver.apple.com.      172762  A       17.254.0.50
; glue
nserver2.apple.com.     172762  A       17.254.0.59
; glue
nserver3.apple.com.     172762  A       17.112.144.50
; glue
nserver4.apple.com.     172762  A       17.112.144.59
; authanswer
www.apple.com.          1762    CNAME   www.apple.com.akadns.net.
; glue
NET.                    172763  NS      A.GTLD-SERVERS.net.
                        172763  NS      B.GTLD-SERVERS.net.
                        172763  NS      C.GTLD-SERVERS.net.
                        172763  NS      D.GTLD-SERVERS.net.
                        172763  NS      E.GTLD-SERVERS.net.
                        172763  NS      F.GTLD-SERVERS.net.
                        172763  NS      G.GTLD-SERVERS.net.
                        172763  NS      H.GTLD-SERVERS.net.
                        172763  NS      I.GTLD-SERVERS.net.
                        172763  NS      J.GTLD-SERVERS.net.
                        172763  NS      K.GTLD-SERVERS.net.
                        172763  NS      L.GTLD-SERVERS.net.
                        172763  NS      M.GTLD-SERVERS.net.
; glue
akadns.NET.             172763  NS      za.akadns.net.
                        172763  NS      zc.akadns.net.
                        172763  NS      zf.akadns.net.
                        172763  NS      zh.akadns.net.
                        172763  NS      eur3.akam.net.
                        172763  NS      use2.akam.net.
                        172763  NS      use4.akam.net.
                        172763  NS      usw5.akam.net.
                        172763  NS      usw6.akam.net.
                        172763  NS      usw7.akam.net.
                        172763  NS      asia3.akam.net.
; authanswer
www.apple.com.akadns.NET. 24    A       17.112.152.32
; glue
za.akadns.NET.          172763  A       208.185.132.176
; glue
zc.akadns.NET.          172763  A       63.241.199.54
; glue
zf.akadns.NET.          172763  A       63.215.198.83
```

```
; glue
zh.akadns.NET.          172763  A       63.208.48.46
; glue
asia3.akam.NET.         172763  A       193.108.154.9
; glue
eur3.akam.NET.          172763  A       193.45.1.103
; glue
use2.akam.NET.          172763  A       63.209.170.136
; glue
use4.akam.NET.          172763  A       80.67.67.182
; glue
usw5.akam.NET.          172763  A       63.241.73.214
; glue
usw6.akam.NET.          172763  A       206.132.100.108
; glue
usw7.akam.NET.          172763  A       65.203.234.27
; glue
A.GTLD-SERVERS.NET.     172763  A       192.5.6.30
; glue
B.GTLD-SERVERS.NET.     172763  A       192.33.14.30
; glue
C.GTLD-SERVERS.NET.     172763  A       192.26.92.30
; glue
D.GTLD-SERVERS.NET.     172763  A       192.31.80.30
; glue
E.GTLD-SERVERS.NET.     172763  A       192.12.94.30
; glue
F.GTLD-SERVERS.NET.     172763  A       192.35.51.30
; glue
G.GTLD-SERVERS.NET.     172763  A       192.42.93.30
; glue
H.GTLD-SERVERS.NET.     172763  A       192.54.112.30
; glue
I.GTLD-SERVERS.NET.     172763  A       192.43.172.30
; glue
J.GTLD-SERVERS.NET.     172763  A       192.48.79.30
; glue
K.GTLD-SERVERS.NET.     172763  A       192.52.178.30
; glue
L.GTLD-SERVERS.NET.     172763  A       192.41.162.30
; glue
M.GTLD-SERVERS.NET.     172763  A       192.55.83.30
; additional
A.ROOT-SERVERS.NET.     604764  A       198.41.0.4
; additional
B.ROOT-SERVERS.NET.     604764  A       192.228.79.201
; additional
C.ROOT-SERVERS.NET.     604764  A       192.33.4.12
; additional
D.ROOT-SERVERS.NET.     604764  A       128.8.10.90
; additional
E.ROOT-SERVERS.NET.     604764  A       192.203.230.10
; additional
F.ROOT-SERVERS.NET.     604764  A       192.5.5.241
; additional
G.ROOT-SERVERS.NET.     604764  A       192.112.36.4
; additional
H.ROOT-SERVERS.NET.     604764  A       128.63.2.53
; additional
```

```
I.ROOT-SERVERS.NET.      604764  A       192.36.148.17
; additional
J.ROOT-SERVERS.NET.      604764  A       192.58.128.30
; additional
K.ROOT-SERVERS.NET.      604764  A       193.0.14.129
; additional
L.ROOT-SERVERS.NET.      604764  A       198.32.64.12
; additional
M.ROOT-SERVERS.NET.      604764  A       202.12.27.33
```

Im Einzelnen lassen sich diese Zwischenergebnisse wie folgt katalogisieren:

- die DNS-Namen und IP-Adressen der 13 Root-Server – zuständig für die Root-Domain . (ein einzelner Punkt),
- die DNS-Namen und IP-Adressen der 13 für die *com* TLD zuständigen DNS-Server,
- die DNS-Namen und IP-Adressen der sechs für die *apple.com*-Domain zuständigen DNS-Server,
- die Information, dass es sich bei */www.apple.com/* um ein Alias für *www.apple.com.akadns.net* handelt,
- die DNS-Namen und IP-Adressen der 13 für die *net* TLD zuständigen DNS-Server,
- die DNS-Namen und IP-Adressen der 11 für die *akadns.net*-Domain zuständigen DNS-Server
- und schließlich die IP-Adresse des Rechners mit dem DNS-Namen *www.apple.com.akadns.net*, nämlich 17.112.152.32.

Ein „optimierter“ DNS-Cache

Interessanterweise verwendet *Internet Sharing* auch einen DNS-Cache, allerdings in einer leicht modifizierten Konfiguration. Sobald der Benutzer *Internet Sharing* aktiviert, wird neben einem DHCP-Server und NAT eben auch BIND als DNS-Cache aktiviert. Statt `/etc/named.conf` wird als Konfigurationsdatei jedoch `/etc/com.apple.named.conf` verwendet:

```
$ ps ax -o command | grep named
/usr/sbin/named -c /etc/com.apple.named.conf.proxy -f
...
```

Diese besondere Konfigurationsdatei legt *Internet Sharing* beim Start automatisch selbst an und löscht sie wieder, sobald es beendet wird. Die Unterschiede zwischen dieser Datei und der im letzten Abschnitt vorgestellten Standardkonfigurationsdatei `/etc/named.conf` sind sehr gering:

```
$ cat /etc/com.apple.named.conf.proxy
// Declares control channels to be used by the rndc utility.
// It is recommended that 127.0.0.1 be the only address used.
// This also allows non-privileged users on the local host to
   manage
// your name server. An empty control set means the utility
   is disabled.
```

```
//
controls {};

options {
        directory "/var/named";
        /*
         * If there is a firewall between you and nameservers you
         * want to talk to, you might need to uncomment the query-
         * source directive below.  Previous versions of BIND
         * always asked questions using port 53, but BIND 8.1 uses
         * an unprivileged port by default.
         */
        // query-source address * port 53;

        listen-on { 192.168.2.1;  };
        forward first;
        forwarders { 192.168.0.1;  };
};

//
// a caching only nameserver config
//
zone "." IN {
        type hint;
        file "named.ca";
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

acl can_query {any;};
```

Im Wesentlichen beschränken sich die Unterschiede zwischen diesen beiden Dateien auf die zusätzlichen Optionen `listen-on`, `forward first` und `forwarders` sowie auf die Anweisung `acl` ganz am Ende der Datei.

Die Option `listen-on` legt dabei einfach fest, dass der DNS-Server Anfragen nur über die interne Netzwerkschnittstelle (in unserem Beispiel mit der IP-Adresse 192.168.2.1) und nicht etwa auch über die externe Schnittstelle akzeptieren wird. Ohne diese Option würde der DNS-Server auf über alle Netzwerkschnittstellen empfangene Anfragen antworten.

Die beiden Optionen `forward` und `forwarders` ändern dagegen das Verhalten des DNS-Servers bei der Bearbeitung von DNS-Anfragen. Ohne diese Option versucht der DNS-Server, ausgehend von einer Gruppe von ihm bekannten Root-Servern, alle DNS-Anfragen selbstständig aufzulösen. Mit Hilfe der `forwarders`-Option lassen sich stattdessen alle Anfragen des DNS-Servers an andere DNS-Server, die *Forwarder*, zur weiteren Bearbeitung wei-

terleiten. Damit ist es z. B. möglich, trotz eines eigenen DNS-Servers den designierten DNS-Server des Internet-Zugangsanbieters zu nutzen. Auf diese Weise können oftmals die Antwortzeiten reduziert werden, wenn der andere DNS-Server über einen größeren und bereits gut gefüllten Cache verfügt.

In unserem Beispiel ist als Forwarder ein interner DNS-Server (192.168. 0.1) eingetragen. Internet Sharing hat hierzu einfach die DNS-Einstellungen der externen Netzwerkschnittstelle übernommen.

Über die Option `forward first` wird der DNS-Server instruiert, bei jeder Anfrage zuerst die Forwarder zu kontaktieren, und nur wenn diese nicht erreichbar sein sollten, eine eigene Bearbeitung zu beginnen. Da in vielen Fällen aber ohnehin keine eigene Bearbeitung möglich ist, wenn die hier eingetragenen DNS-Server nicht erreichbar sind – z. B. weil der Internet-Zugangsanbieter nicht erreichbar ist – könnte man an dieser Stelle auch `forward only` statt `forward first` angeben. Auch mit `forward only` werden alle Anfragen an die Forwarder weitergeleitet. Sollten diese aber nicht erreichbar sein, gibt der DNS-Cache sofort eine Fehlermeldung zurück, ohne eine eigene Bearbeitung zu versuchen.

Die letzte Anweisung, `acl`, hat keine praktische Auswirkung. Mit dieser Anweisung werden Gruppen von Rechnern definiert und mit einem Namen versehen. Die Anweisung `acl can_query {any;};` definiert eine Gruppe von Rechnern mit dem Namen `can_query`, die alle Rechner enthält. Eine solche Gruppe könnte man mit Hilfe der (in der von Internet Sharing nicht erzeugten) Option `allow-query` dazu verwenden, den Zugriff auf den DNS-Server nur auf eine bestimmte Gruppe von Rechnern einzuschränken, wenn man die Konfigurationsdatei wie folgt modifizieren würde:

```
...
options {
...
allow-query { can_query; };
...
}
...
acl can_query { 192.168.2.0/24; };
...
```

Auf diese Weise könnte man den Zugriff auf Hosts mit IP-Adressen aus dem Netzwerk 192.168.2.0/24, d. h. 192.168.2.1 bis 192.168.2.254 einschränken.

Änderungen sind allerdings an der von Internet Sharing verwendeten Konfigurationsdatei nicht möglich, weil diese fest in das dazugehörende Steuerungsprogramm `/usr/libexec/InternetSharing` hineinkompiliert ist. Es steht allerdings nichts dem Ansinnen im Wege, einige der in diesem Abschnitt dargestellten Optionen in `/etc/named.conf` einfach zu übernehmen.

Dies könnte dann einfach so aussehen:

```
// Steuerung mit rndc soll nur lokal und über den Port 54
// möglich sein

controls {
```

```
     inet 127.0.0.1 port 54 allow {any; };
};

options {

        // Verwende /var/named als Arbeitsverzeichnis
        directory "/var/named";

        // Akzeptiere Anfragen nur über die interne
        // Netzwerkschnittstelle
        listen-on { 192.168.2.1;  };

        // Bearbeite die Anfragen niemals selbst, sondern
        // leite sie an die Forwarder weiter
        forward only;

        // Verwende 192.168.0.1 als Forwarder
        forwarders { 192.168.0.1;  };

        // Akzeptiere Anfragen nur von den Hosts aus der
        // can_query Gruppe
        allow-query { can_query; };
};

// Root-Server
zone "." IN {
        type hint;
        file "named.ca";
};

// Localhost <-> 127.0.0.1

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

// Definiere Gruppe can_query

acl can_query {192.168.2.0/24;};
```

Um diese Konfigurationsdatei zu verwenden, müssen die IP-Adresse der internen Netzwerkschnittstelle (Option `listen-on`), die IP-Adresse des als Forwarder zu verwendenden DNS-Servers (Option `forwarders`) und die IP-Adressen der als Absender von DNS-Anfragen zugelassenen Hosts (Anweisung `acl`) an das eigene Netzwerk angepasst werden.

Ein privater DNS-Namensraum

Normalerweise ist die Übernahme der Verantwortung für einen Teil des öffentlichen DNS-Namensraums mit einem nicht unbeträchtlichen Aufwand ver-

bunden. Das hängt zum einen mit dem formalen Registrierungsverfahren, und zum anderen mit den technischen Anforderungen an den Betrieb von öffentlichen DNS-Servern zusammen.

Hat man diese Verantwortung übernommen, muss man rund um die Uhr gewährleisten, dass alle diesen Namensraum betreffenden Anfragen korrekt beantwortet werden können. Das wiederum stellt hohe Anforderungen an die Korrektheit der DNS-Server-Konfiguration und an die Konsistenz der verwalteten Daten. Hinzu kommt, dass man dabei in der Regel mindestens zwei DNS-Server in Betrieb nehmen muss, um gegen Serverausfälle vorzusorgen.

Während der erste DNS-Server, der so genannte *Master*, die den verwalteten DNS-Namensraum betreffenden Daten einer Konfigurationsdatei entnimmt, wird der zweite (und ggf. jeder weitere) DNS-Server normalerweise als so genannter *Slave* konfiguriert und bezieht diese Daten regelmäßig vom Master. Auf diese Weise muss man bei Änderungen im DNS-Namensraum (z. B. Neueintragungen, Umbenennungen oder Löschungen von Hosts) nur die Konfigurationsdateien eines einzigen DNS-Servers editieren – und automatische Replikation ist regelmäßig, wenn sie denn einmal richtig konfiguriert wurde, viel weniger fehleranfällig als die wiederholte Replikation per Hand.

Falsche Konfiguration kann Störungen hervorrufen, die nicht nur das eigene, sondern auch fremde Netzwerke betreffen. Von Experimenten in diesem Bereich ist daher abzuraten, solange man sich nicht ganz sicher ist, was man tut.

Weit ungefährlicher ist die Einrichtung eines privaten DNS-Servers in einem privaten Netzwerk, welcher lediglich von internen Hosts angesprochen werden kann. Solange ein solcher DNS-Server der Außenwelt unbekannt ist, sind die Konfigurationsfehler in ihren Auswirkungen auf das lokale Netzwerk begrenzt und die Risiken daher überschaubarer. Natürlich können externe Hosts dann auch nicht diese internen DNS-Namen verwenden.

Ein privater DNS-Server unterscheidet sich nur wenig von einem reinen DNS-Cache. Der wesentliche Unterschied liegt darin begründet, dass ein DNS-Cache in gewissem Sinne „dumm“ ist: er kann fast keinerlei Anfragen selbst beantworten, sondern muss immer andere DNS-Server fragen, die ihm dann die Antworten liefern müssen. Seine „Intelligenz“ beschränkt sich auf sein „Gedächtnis“ – der DNS-Cache merkt sich für eine gewisse Zeit (engl. time-to-live, kurz TTL) die erhaltenen Antworten, so dass er nicht bei jeder Frage erneut selbst nachfragen muss.

Ein privater DNS-Server verhält sich ähnlich, hat aber selbst ein gewisses Wissen, so dass er bestimmte Anfragen selbst, also ohne fremde Hilfe, beantworten kann. Dieses Wissen bezieht er aus zwei Tabellen: einer Zuordnung von DNS-Namen zu IP-Adressen und einer Zuordnung von IP-Adressen zu DNS-Namen.

Den Teil des DNS-Namensraums, den eine solche Tabelle abdeckt, bezeichnet man als eine Zone. Um also einen DNS-Cache in einen privaten DNS-Server zu verwandeln, muss man zwei Zonen hinzufügen: die erste für

die Zuordnung von DNS-Namen zu den privaten IP-Adressen und eine zweite für die Zuordnung von IP-Adressen zu DNS-Namen.

Um diese beiden Zonen zu definieren, muss man einen Teil des DNS-Namensraums auswählen, den man mit dem privaten DNS-Server abdecken möchte. Hat das private Netzwerk keine Verbindung zur Außenwelt, und ist eine solche Anbindung auch für die Zukunft nicht geplant, hat man hierfür freie Auswahl. Da wegen der völligen Abschottung Überschneidungen mit offiziellen DNS-Namen folgenlos bleiben, könnte als Namensraum sogar *apple.com* o.Ä. gewählt werden.

Eine solche völlige Abschottung findet heutzutage aber nur extrem selten statt und ist definitiv als rein hypothetisch zu betrachten. Der private DNS-Namensraum muss daher immer so gewählt werden, dass die Kommunikation mit der Außenwelt nicht behindert wird. Würde man einen DNS-Namensraum wählen, der bereits offiziell verwendet wird, also z. B. die Domain *apple.com*, würden alle zu dieser Domain gehörenden Hosts vom privaten Netzwerk nicht mehr erreichbar sein. Anfragen nach DNS-Namen aus diesem dann eigenen DNS-Namensraum würde der DNS-Server nämlich direkt mit dem Wissen aus seinen Konfigurationsdateien beantworten. Das schließt auch Fehlermeldungen ein, wenn der angefragte Name nicht definiert wurde. Eine weitere Bearbeitung der Anfrage mit Hilfe der öffentlichen DNS-Server würde nicht stattfinden.

Um dieses Problem zu umgehen, gibt es zwei Lösungsvarianten: entweder man verwendet eine bisher nicht genutzte Top-Level-Domain oder eine Domain, die man selbst registriert hat.

Im ersten Fall denkt man sich eine neue Top-Level-Domain aus, z. B. *meine-firma*. Solange man sich sicher sein kann, dass eine offizielle TLD mit dem gleichen Namen nicht existiert, kann der private DNS-Server alle diese Domain betreffenden Anfragen abfangen, ohne irgendwelche Probleme zu verursachen. Probleme können sich dann ergeben, wenn neue TLD zur offiziellen Nutzung freigegeben werden.

Da Namensdefinitionen in der Regel langlebig sind, kann diese Variante langfristig Probleme verursachen. In der Vergangenheit hatten einige Netzwerkadministratoren z. B. entschieden, den verhältnismäßig nahe liegenden Domain-Namen *local* als private TLD zu wählen. Unglücklicherweise führt diese Entscheidung zu Kompatibilitätsproblemen mit Bonjour/ZeroConf bzw. Multicast DNS (mDNS). Bonjour/ZeroConf verwendet die TLD *local* um zwischen Multicast DNS und dem herkömmlichen (Unicast) DNS zu unterscheiden. Ein entsprechender Resolver sendet normalerweise alle Anfragen, die Namen mit der Endung *.local* betreffen, ausschließlich an die mDNS-Multicast-Adresse 224.0.0.251, so dass die in der privaten TLD definierten Rechner gar nicht mehr über ihre Namen erreicht werden.

Gänzlich umgehen lässt sich diese Problematik, indem man die zweite Variante nimmt und eine Domain wählt, die man bereits selbst registriert hat, also z. B. *meine-firma.de*. Um den Konfigurationsaufwand zu minimieren,

sollte als privater DNS-Namensraum eine Unterdomain (z. B. *intern.meine-firma.de*) dieser öffentlichen Domain verwendet werden. Auf diese Weise fühlt sich der private DNS-Server dann auch ausschließlich für diese Unterdomain zuständig und bearbeitet Anfragen, welche die öffentliche Domain betreffen, korrekt mit Hilfe der öffentlichen DNS-Server.

Auf diese Weise werden auch Anfragen nach den in der öffentlichen Domain eventuell bereits definierten Hosts, wie beispielsweise Web-, Mail- oder FTP-Server (etwa www.meine-firma.de, mail.meine-firma.de oder ftp.meine-firma.de) korrekt beantwortet.

Um eine Zone zu definieren, muss man eine entsprechende Konfigurationsdatei – die Zonen-Datei – erstellen. Für einen privaten DNS-Server benötigen wir zwei solche Zonen-Dateien. Die erste Zonen-Datei enthält als wesentlichen Bestandteil eine Tabelle von DNS-Namen zusammen mit den dazugehörigen IP-Adressen. Auf der Grundlage dieser ersten Tabelle kann unser DNS-Server Anfragen nach der zu einem bestimmten Namen passenden IP-Adresse beantworten. Die zweite Zonen-Datei enthält umgekehrt eine Tabelle von IP-Adressen zusammen mit den dazugehörigen DNS-Namen. Diese Datei wird für die Beantwortung von Anfragen nach dem zu einer IP-Adresse passenden DNS-Namen verwendet.

Eine Zonen-Datei besteht aus einer Reihe von *Resource Records*, den DNS-Datensätzen, die den Inhalt der lokalen DNS-Datenbank des betreffenden DNS-Servers konstituieren. Es gibt eine Vielzahl von Datensatztypen; die wichtigsten – und für einen privaten DNS-Server ausreichenden – Typen lauten:

Typ-SOA-Datensatz (engl. *start of authority*): Mit diesem Datensatz wird der DNS-Server für den in der Zone-Datei definierten Teil des DNS-Raums für zuständig erklärt. Ein privater DNS-Server weiß somit, dass er keinen anderen DNS-Server nach diesen DNS-Namensraum betreffenden Angaben fragen muss. Jede Zonen-Datei enthält genau einen SOA-Datensatz.

Typ-NS-Datensatz (engl. *name server*): Mit diesem Datensatz werden die für diesen Teil des DNS-Namensraums zuständigen DNS-Server deklariert. Auf diese Weise reicht es bereits, die IP-Adresse eines einzigen für eine Domain zuständigen DNS-Servers zu kennen, um an die Liste aller zuständigen DNS-Server zu kommen (um dann denjenigen mit den schnellsten Antwortzeiten auszuwählen). Normalerweise enthalten Zonen-Dateien öffentlicher DNS-Server immer mehrere NS-Datensätze, da es mehrere zuständige DNS-Server gibt (in der Regel konfiguriert als ein Master und mehrere Slaves).

Typ-A-Datensatz (engl. *address*): Dieser Datensatz bildet einen DNS-Namen auf eine IP-Adresse ab. Normalerweise gibt es in einer Zone-Datei genauso viele A-Datensätze wie Hosts mit zugewiesenen IP-Adressen.

Typ-PTR-Datensatz (engl. *pointer*): Dieser Datensatz bildet eine IP-Adresse auf einen DNS-Namen ab. Auch von diesen Datensätzen gibt

es in einer Zone-Datei normalerweise genauso viele wie Hosts mit zugewiesenen IP-Adressen.

Typ-CNAME-Datensatz (engl. *canonical name*): Diese Datensätze definieren Aliase, indem sie von dem Alias auf den kanonischen („echten“) Namen verweisen. Aliase werden häufig verwendet, um zusätzlich zu den kanonischen Namen für Benutzer leicht zu merkende Rechnernamen zu definieren. Kanonische Namen werden häufig nach einem festen Schema vergeben – Aliase orientieren sich an den auf den Rechnern aktuell betriebenen Diensten. Sehr häufig verwendet werden Aliase wie z. B. *www*, *mail* oder *ftp*[17].

Die Datei beginnt mit einer Festlegung des Gültigkeitszeitraums (engl. time to live, kurz TTL). Diese Angabe legt fest, wie lange ein anderer DNS-Server die Angaben zu dieser Zone zwischenspeichern darf. Nach Ablauf dieses Zeitraums muss erneut bei unserem DNS-Server nachgefragt werden, um eventuelle Änderungen nicht zu verpassen.

Der Gültigkeitszeitraum wird normalerweise in Sekunden angegeben. Ein typischer Wert wäre z. B. drei Stunden und müsste mit der folgenden Anweisung deklariert werden:

```
$TTL 10800
```

Ein sehr kurzer TTL-Wert hat zur Folge, dass andere Server immer wieder bei unserem DNS-Server nachfragen müssen, ob sich was geändert hat. Erfolgen Änderungen sehr selten, kann durch eine Erhöhung des TTL-Werts die Netzwerkbelastung durch solche Anfragen minimiert werden. Ein sehr großer Wert hat den Nachteil, dass die Verbreitung von Änderungen länger dauert. Für einen einzelnen privaten DNS-Server spielen diese Überlegungen aber keine Rolle.

Als nächstes muss ein SOA-Datensatz angegeben werden[18]. Ein SOA-Datensatz, der die Zuständigkeit unseres DNS-Servers für die Domain *intern.meine-firma.de* erklärt kann, wie folgt definiert werden:

```
intern.meine-firma.de. IN  SOA server.intern.meine-firma.de.
jemand.meine-firma.de. (
       1      ;  Seriennummer (engl. serial number)
       3h     ; Aktualisierungsintervall (engl. refresh)
       1h     ; Wiederholungsintervall (engl. retry)
       1w     ; Ablaufintervall (engl. expire)
       1h )   ; Negatives TTL-Intervall
```

Der SOA-Datensatz beginnt mit der Angabe der zu verwaltenden Domain (im Beispiel *intern.meine-firma.de.* – der Punkt am Ende darf hier nicht

[17] Weiter oben haben wir gesehen, dass *www.apple.com* ein in der *apple.com*-Zone definiertes Alias auf einen Rechner mit dem kanonischen Namen *www.apple.com.akadns.net* darstellt.

[18] Die Reihenfolge der Datensätze spielt eigentlich keine Rolle – eingebürgert hat sich jedoch die hier angegebene Reihenfolge.

weggelassen werden!). Mit `IN` ist Internet gemeint, und es bezeichnet die Datensatz-Klasse für TCP/IP-basierte Netzwerke – aus historischen Gründen unterstützt BIND noch zwei weitere Datensatz-Klassen (für Chaosnet- und Hesiod-Netzwerke), die heutzutage aber kaum mehr gebräuchlich sind. Das Schlüsselwort `SOA` steht für den Datensatztyp.

Als nächstes folgt der vollständige DNS-Name des zuständigen DNS-Servers (im Beispiel *server.intern.meine-firma.de.* gefolgt von der Email-Adresse des verantwortlichen Administrators. Anders als normalerweise im Email-Verkehr üblich, wird an dieser Stelle statt des Zeichens „@“ ein Punkt zwischen dem Benutzer- und Domain-Namen verwendet (im Beispiel *jemand.meine-firma.de.*, zu interpretieren als *jemand@meine-firma.de.*).

Die Seriennummer identifiziert eindeutig eine bestimmte Version der Zonen-Datei. Sie wird von Slave-Servern benutzt, um Aktualisierungen zu erkennen, und sollte daher bei jeder Änderung erhöht werden. Auch wenn man keine Slave-Server betreibt, ist es nicht falsch, wenn man sich diesen Schritt von Anfang an angewöhnt. Die Nummer an sich hat keine weitere Bedeutung und kann frei gewählt werden. Man kann, wie im Beispiel, einfach bei 1 beginnen und von da an fortschreiten – man kann aber auch das Änderungsdatum zusammen mit einer Änderungsnummer im Format JJJJMMTTNN kodieren, z. B. 2004073001.

Auch Aktualisierungs-, Wiederholungs- und Ablaufintervall werden ausschließlich von Slave-Servern verwendet. Mit dem Aktualisierungsintervall legt man fest, wie oft ein Slave-Server seine Daten aktualisieren wird. Das Wiederholungsintervall gibt an, wie oft ein Slave-Server versuchen soll, den Master-Server zu erreichen, falls dieser einmal ausfallen sollte. Das Ablaufintervall gibt schließlich an, wann ein Slave seine Daten für ungültig zu erklären hat, wenn der Master längere Zeit unerreichbar bleibt.

Im Beispiel wird also festgelegt, dass ein Slave-Server alle drei Stunden seine Daten aktualisieren muss. Sollte der Master-Server mal ausfallen, muss der Slave-Server stündlich versuchen, ihn doch noch zu erreichen. In der Zwischenzeit darf der Slave Anfragen weiterhin aus seinem Zwischenspeicher bedienen, solange der letzte Kontakt mit dem Master nicht länger als eine Woche zurückliegt.

Für einen DNS-Server ohne Slave-Server sind diese Werte natürlich ohne praktische Bedeutung. Anders der letzte Wert – das Negative TTL-Intervall. Dieser Wert gibt an, wie lange negative Antworten (z. B. Name existiert nicht) ihre Gültigkeit behalten. Hier gelten die gleichen Bedingungen wie bei der Auswahl des TTL-Werts weiter oben – ein langes Intervall minimiert die Zahl der Anfragen anderer DNS-Server, ein kurzes Intervall erhöht die Verbreitungsgeschwindigkeit von Änderungen. Allerdings spielt auch dieser Wert bei einzelnen privaten DNS-Servern keine Rolle.

Die im Beispiel angegebenen Beispielwerte können daher ohne weiteres für einen einzelnen privaten DNS-Server einfach übernommen werden.

Nach dem SOA-Datensatz werden normalerweise mehrere NS-Datensätze definiert. Jeder NS-Datensatz deklariert einen, für die jeweilige Domain zuständigen DNS-Server. Will man nur einen einzelnen DNS-Server (ohne Slaves) betreiben, gibt es natürlich auch nur einen NS-Datensatz:

```
intern.meine-firma.de. IN NS server.intern.meine-firma.de.
```

Der NS-Datensatz beginnt mit dem Domain-Namen, gefolgt von der Datensatz-Klasse (IN) und dem Datensatztyp (NS). Es folgt der DNS-Name eines DNS-Servers.

Den Kern der Zonen-Datei bilden die Abbildungs-Datensätze. In einer regulären Zonen-Datei handelt es sich dabei um Datensätze von Typ A (Abbildung von DNS-Namen auf IP-Adressen), in einer Zonen-Datei für eine *in-addr.arpa* Unter-Domain sind es Datensätze vom Typ PTR (Abbildung von IP-Adressen auf DNS-Namen).

Die Abbildungsdatensätze vom Typ A sehen wie folgt aus:

```
host1.intern.meine-firma.de.   IN A    192.168.0.1
host2.intern.meine-firma.de.   IN A    192.168.0.2
host3.intern.meine-firma.de.    IN A    192.168.0.3
host4.intern.meine-firma.de.    IN A    192.168.0.4
host5.intern.meine-firma.de.    IN A    192.168.0.5
host6.intern.meine-firma.de.    IN A    192.168.0.6
host7.intern.meine-firma.de.    IN A    192.168.0.7
host8.intern.meine-firma.de.    IN A    192.168.0.8
host9.intern.meine-firma.de.    IN A    192.168.0.9
host10.intern.meine-firma.de.   IN A    192.168.0.10
```

Jeder A-Datensatz beginnt mit dem DNS-Namen eines Hosts (im Beispiel natürlich nicht besonders einfallsreich). Es folgen die übliche Klassenkennzeichnung (IN) und der Datensatztyp (A), gefolgt von der IP-Adresse des Hosts.

Als Ergänzung zu A-Datensätzen werden oft zusätzlich Datensätze vom Typ CNAME verwendet. Mit deren Hilfe lassen sich Aliase definieren, wie z. B.:

```
server.intern.meine-firma.de.   IN CNAME   host1.intern.mei-
ne-firma.de.
```

Auf diese Weise lässt sich der Rechner mit dem DNS-Namen *host1* auch über das Alias *server* ansprechen.

Die Abbildungsdatensätze vom Typ PTR sehen hingegen so aus:

```
1.0.168.192.in-addr.arpa.     IN PTR    host1.example.com.
2.0.168.192.in-addr.arpa.     IN PTR    host2.example.com.
3.0.168.192.in-addr.arpa.     IN PTR    host3.example.com.
4.0.168.192.in-addr.arpa.     IN PTR    host4.example.com.
5.0.168.192.in-addr.arpa.     IN PTR    host5.example.com.
6.0.168.192.in-addr.arpa.     IN PTR    host6.example.com.
7.0.168.192.in-addr.arpa.     IN PTR    host7.example.com.
8.0.168.192.in-addr.arpa.     IN PTR    host8.example.com.
9.0.168.192.in-addr.arpa.     IN PTR    host9.example.com.
10.0.168.192.in-addr.arpa.     IN PTR    host10.example.com.
```

Die vollständige Zonen-Datei für die Domain *intern.meine-firma.de* könnte demnach folgendes Aussehen haben:

```
$TTL 10800

intern.meine-firma.de. IN  SOA server.intern.meine-firma.de.
jemand.meine-firma.de. (
       1      ;  Seriennummer (engl. serial number)
       3h     ; Aktualisierungsintervall (engl. refresh)
       1h     ; Wiederholungsintervall (engl. retry)
       1w     ; Ablaufintervall (engl. expire)
       1h )   ; Negatives TTL-Intervall

intern.meine-firma.de. IN NS server.intern.meine-firma.de.

host1.intern.meine-firma.de.   IN A    192.168.0.1
host2.intern.meine-firma.de.   IN A    192.168.0.2
host3.intern.meine-firma.de.    IN A    192.168.0.3
host4.intern.meine-firma.de.    IN A    192.168.0.4
host5.intern.meine-firma.de.    IN A    192.168.0.5
host6.intern.meine-firma.de.    IN A    192.168.0.6
host7.intern.meine-firma.de.    IN A    192.168.0.7
host8.intern.meine-firma.de.    IN A    192.168.0.8
host9.intern.meine-firma.de.    IN A    192.168.0.9
host10.intern.meine-firma.de.   IN A    192.168.0.10

server.intern.meine-firma.de.    IN CNAME
host1.intern.meine-firma.de.
```

Die entsprechende Zonen-Datei für die Domain *0.168.192.in-addr.arpa* könnte hingegen so aussehen:

```
$TTL 10800

0.168.192.in-addr.arpa. IN SOA server.intern.meine-firma.de.
jemand.meine-firma.de. (
       1      ;  Seriennummer (engl. serial number)
       3h     ; Aktualisierungsintervall (engl. refresh)
       1h     ; Wiederholungsintervall (engl. retry)
       1w     ; Ablaufintervall (engl. expire)
       1h )   ; Negatives TTL-Intervall

0.168.192.in-addr.arpa. IN NS server.intern.meine-firma.de.

1.0.168.192.in-addr.arpa.     IN PTR    host1.example.com.
2.0.168.192.in-addr.arpa.     IN PTR    host2.example.com.
3.0.168.192.in-addr.arpa.     IN PTR    host3.example.com.
4.0.168.192.in-addr.arpa.     IN PTR    host4.example.com.
5.0.168.192.in-addr.arpa.     IN PTR    host5.example.com.
6.0.168.192.in-addr.arpa.     IN PTR    host6.example.com.
7.0.168.192.in-addr.arpa.     IN PTR    host7.example.com.
8.0.168.192.in-addr.arpa.     IN PTR    host8.example.com.
9.0.168.192.in-addr.arpa.     IN PTR    host9.example.com.
10.0.168.192.in-addr.arpa.     IN PTR    host10.example.com.
```

Die beiden Zonendateien müssen als Textdateien angelegt und in einem für den DNS-Server zugänglichen Verzeichnis abgelegt werden – unter Mac OS X ist dafür das Verzeichnis `/var/named/` vorgesehen. Die Namen der Zonendateien können frei gewählt werden, sollten jedoch am besten

einen Rückschluss auf die in den Dateien definierten Domains erlauben. Apple selbst verwendet in Mac-OS-X-Servern als Dateinamen einfach den Domainnamen mit der Endung *.zone*. Analog könnten wir unsere Zone-Dateien *intern.meine-firma.de.zone* und *0.168.192.in-addr.arpa.zone* nennen.

Damit die Zonen-Dateien vom DNS-Server auch wirklich geladen und verwendet werden, muss man sie in der Konfigurationsdatei `/etc/named.conf` mit Hilfe der Anweisung `zone` deklarieren. Als Ausgangsbasis kann man einfach die oben definierte Konfigurationsdatei für einen DNS-Cache verwenden und sie um zwei Zone-Anweisungen ergänzen:

```
// Steuerung mit rndc soll nur lokal und über den Port 54
// möglich sein

controls {
     inet 127.0.0.1 port 54 allow {any; };
};

options {

        // Verwende /var/named als Arbeitsverzeichnis
        directory "/var/named";

        // Akzeptiere Anfragen nur über die interne
        // Netzwerkschnittstelle
        listen-on { 192.168.2.1;  };

        // Bearbeite die Anfragen niemals selbst, sondern
        // leite sie an die Forwarder weiter
        forward only;

        // Verwende 194.25.2.129 als Forwarder
        forwarders { 194.25.2.129;  };

        // Akzeptiere Anfragen nur von den Hosts aus der
        // can_query Gruppe
        allow-query { can_query; };
};

// Root-Server
zone "." IN {
        type hint;
        file "named.ca";
};

// localhost <-> 127.0.0.1

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};
```

```
// intern.meine-firma.de <-> 192.168.0.0/24

zone "intern.meine-firma.de" IN {
        type master;
        file "intern.meine-firma.de.zone";
};

zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "0.168.192.in-addr.arpa.zone";
};

// Definiere Gruppe can_query

acl can_query {192.168.2.0/24;};
```

Diese modifizierte Konfigurationsdatei unterscheidet sich von der DNS-Cache-Version lediglich durch zwei `zone`-Anweisungen. Die erste zusätzliche Anweisung lautet:

```
zone "intern.meine-firma.de" IN {
        type master;
        file "intern.meine-firma.de.zone";
};
```

Sie entspricht der ersten zu ladenden Zone-Datei, nämlich `intern.meine-firma.de.zone`. Die zweite zusätzliche `zone`-Anweisung lautet:

```
zone "0.168.192.in-addr.arpa" IN {
        type master;
        file "0.168.192.in-addr.arpa.zone";
};
```

Diese Anweisung entspricht wiederum der zweiten zu ladenden Zone-Datei `0.168.192.in-addr.arpa.zone`. Das Attribut `type master` legt dabei fest, dass die Zonen-Daten aus der angegeben Datei zu lesen sind; das Attribut `file` deklariert den Namen der Zonen-Datei. Weitere Attribute sind nicht notwendig – das Attribut `allow-update`, welches bei der Deklaration der Root-Zone, der localhost-Zone und der 0.0.127.in.addr.arpa-Zone angegeben ist, deaktiviert explizit so genannte DNS Dynamic Updates, was aber dem Standard-Verhalten des DNS-Servers entspricht und daher überflüssig ist.

3.5 Auf freigegebene Verzeichnisse zugreifen

Der gemeinsame Zugriff auf Dateien über Rechnergrenzen hinweg ist eine der Hauptanwendungen von Rechnernetzen. Dieser Zugriff erfolgt in der Regel nach dem Client-Server-Prinzip, wobei ein Rechner die Rolle des Servers und ein anderer die Rolle des Clients übernimmt.

Der Server-Rechner führt ein Server-Programm aus, welches ausgewählte lokale Verzeichnisse dieses Rechners für geeignete Clients *exportiert* bzw. *freigibt*.

Der Client-Rechner verfügt über Software, welche neben dem Zugriff auf lokal gespeicherte Dateien zusätzlich auch den Zugriff auf entfernte, auf an-

deren Rechnern gespeicherte Dateien erlaubt. Dazu *importiert* der Client die vom Server-Rechner exportierten Verzeichnisse und präsentiert sie dem Anwender entweder als Teil der lokalen Verzeichnishierarchie oder als separate, externe (Netzwerk-)Datenträger.

Moderne Betriebssysteme wie Mac OS X verfügen in der Regel sowohl über die für den gemeinsamen Dateizugriff notwendige Client- als auch die Server-Software. Mac-OS-X-Rechner können in diesem Zusammenhang also grundsätzlich sowohl die Client- als auch die Serverrolle übernehmen – auch gleichzeitig.

Damit die Kommunikation und damit der Dateizugriff erfolgreich ablaufen können, müssen Client und Server ein gemeinsames Kommunikationsprotokoll verwenden. Im Gegensatz zum klassischen Mac OS, welches ohne Zusatzsoftware mit AFP nur ein einziges Kommunikationsprotokoll beherrschte, stehen in Mac OS X mindestens fünf verschiedene Protokolle zur Verfügung:

AppleTalk Filing Protocol (kurz AFP) ist ein von Apple Computer entwickeltes universelles Protokoll für Macintosh-, MSDOS-, Windows- und Unix-Systeme;

Server Message Block (kurz SMB) ist ein von Microsoft entwickeltes Protokoll, ursprünglich ausschließlich für MS-DOS- und Windows-Systeme – SMB wurde 1996 unter dem Namen Common Internet File System (kurz CIFS) standardisiert;

Network File System (kurz NFS) ist ein von Sun Microsystems entwickeltes Protokoll und De-facto-Standard bei Unix-Systemen;

File Transfer Protocol (kurz FTP) ist ein offener Standard für Dateiübertragung in TCP/IP-Netzwerken;

Web-based Disributed Authoring and Versioning (kurz WebDAV) ist eine Erweiterung des HTTP-Protokolls um Dateizugriffs-, -bearbeitungs und -verwaltungsfunktionen.

Verbindungen zu Servern, deren Name oder IP-Adresse bereits bekannt ist, können manuell entweder mit Hilfe des Finder-Dialogs „Mit Server verbinden“ (Menübefehl „Mit Server verbinden“ im Menü „Gehe zu“ oder die Tastenkombination Befehl-K) (Abb. 3.29) oder mit einem der folgenden Befehle aufgebaut werden:

```
mount_afp
mount_smbfs
mount_nfs
mount_ftp
mount_webdav
```

Will man mit Hilfe des Finder-Dialogs eine Verbindung zu einem AFP-Server, der AFP über TCP unterstützt, aufbauen, genügt die Angabe des DNS-Namens oder der IP-Adresse des Servers. In allen anderen Fällen muss man eine URL (engl. Uniform Resource Locator) als „Server-Adresse“ angeben und damit zumindest auch das Protokoll festlegen.

Abb. 3.29. Verbindungen zu Servern, deren Name oder IP-Adresse bereits bekannt ist, können einfach mit Hilfe des Dialogs „Mit Server verbinden“ aufgebaut werden.

Abb. 3.30. In den Finder-Voreinstellungen lässt sich festlegen, ob Netzwerkdatenträger in der Seitenleiste der Finder-Fenster angezeigt werden.

Eine URL setzt sich allgemein aus einem Protokollidentifikator und dem DNS-Namen bzw. der IP-Adresse des Serverrechners zusammen. Sie kann optional mit einer Pfadangabe enden, z. B.

```
afp://server.example.com/admin
```

Die hier als Beispiel verwendete URL adressiert das Verzeichnis `admin` auf dem Server `server.example.com`, auf den mit AFP über TCP zugegriffen werden kann. Die Angabe des Verzeichnisses ist optional – fehlt sie, präsentiert der Finder dem Benutzer eine Auswahlliste aller auf dem angegebenen Server freigegebenen Verzeichnisse.

Neben AFP URLs unterstützt der Finder an dieser Stelle auch FTP (`ftp://...`), WebDAV (`http://...`), NFS (`nfs://...`) und SMB (`smb://...`) URLs. Das allgemeine Schema lautet:

```
protokoll://server-name/pfad
```

Zusätzlich können optional auch ein Benutzername und ein Passwort für die Authentifizierung angegeben werden:

```
protokoll://name:passwort@server-name/pfad
```

Im Allgemeinen sollte man Passwörter in URLs jedoch vermeiden, da die URLs als Klartext, d. h. unverschlüsselt übertragen werden, so dass die Passwörter leicht abgehört werden könnten.

Die an sich komfortable Liste der zuletzt benutzten Server, die der Dialog „Mit Server verbinden“ selbsttätig fortschreibt, birgt in diesem Fall ebenso ein zusätzliches Risiko wie die Finder-Voreinstellungsdatei `com.apple.finder.plist`, in der die zuletzt benutzte URL unverschlüsselt abgelegt wird.

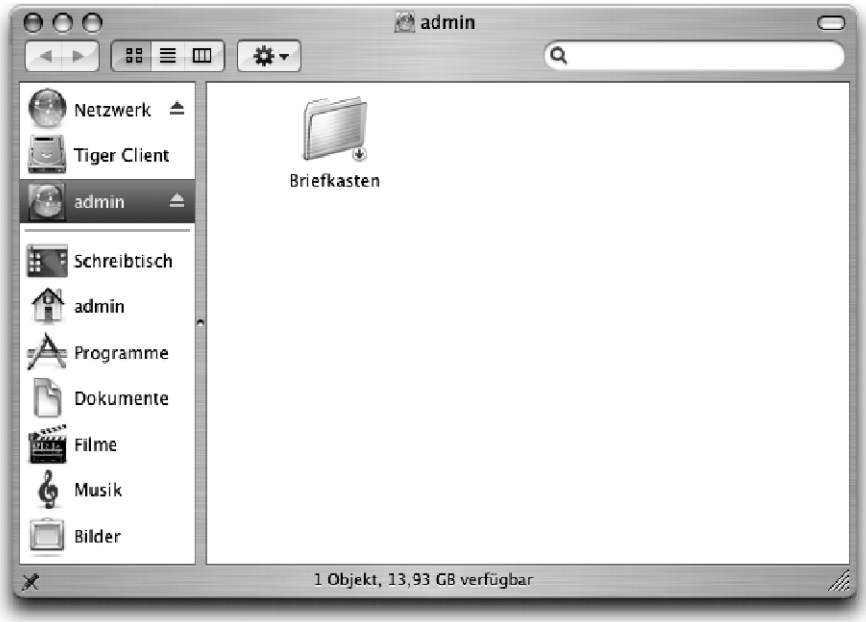


Abb. 3.31. Das Icon des Netzwerk-Datenträgers „admin“ wird in der Seitenleiste des Finders zusammen mit einem Auswurf-Symbol angezeigt.

Bei Verbindungen, die manuell über den Finder hergestellt werden, kann man auf die Angabe des Benutzernamens und des Passworts auch ruhig verzichten, da der Finder nach Eingabe der URL selbstständig diese Angaben in einem weiteren Dialog nachfordert. Dadurch erspart man sich leicht den einen oder anderen Tippfehler.

Für die Darstellung von aktivierten Netzwerk-Datenträgern verwendet der Finder die gleiche Metapher wie für andere externe Datenträger, wie z. B. CDs, DVDs oder externe (beispielsweise über FireWire oder USB angeschlossene) Festplatten.

Das Icon des Netzwerk-Datenträgers wird deshalb normalerweise in der Seitenleiste des Finders zusammen mit einem Auswurf-Symbol angezeigt (Abb. 3.31). Durch einen Klick auf dieses Auswurf-Symbol oder die Auswahl des Befehls „Auswerfen“ aus dem Ablage-Menü kann die Verbindung zum Netzwerk-Datenträger jederzeit wieder getrennt werden.

Ebenso wie bei externen Datenträgern legt der Finder deshalb auch für jeden aktivierten Netzwerk-Datenträger im unsichtbaren Verzeichnis `/Volumes` ein leeres Verzeichnis an und montiert dort das Dateisystem dieses Datenträgers. Dieses Verzeichnis ist nur für den Finder-Benutzer unsichtbar:

```
$ ls -l /Volumes/
total 8
```

Abb. 3.32. In den allgemeinen Finder-Voreinstellungen legt der Benutzer fest, ob er Netzwerkdatenträger auf dem Schreibtisch sehen will oder nicht.

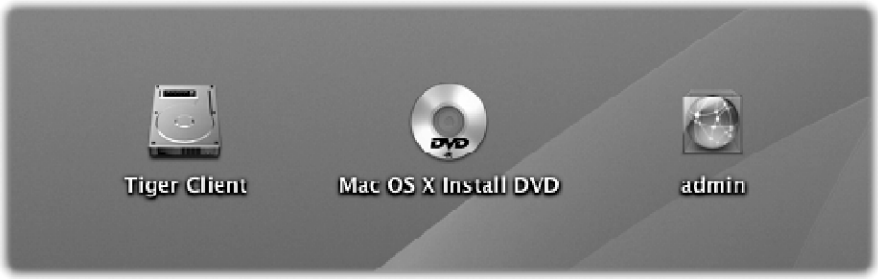


Abb. 3.33. Der Finder kann lokale Datenträger, Wechselmedien und aktivierte Netzwerkdatenträger als Icons auf dem Schreibtisch darstellen.

```
lrwxr-xr-x   1 root   admin    1 Jul 24 13:02 Tiger Client -> /
dr-xr-xr-x   4 admin  staff  264 Jul  9 14:33 admin
```

Das Beispiel zeigt den Netzwerk-Datenträger „admin“ als Unterverzeichnis von `/Volumes`. Im Finder wird „admin“ als Symbol in der Seitenleiste des Finder-Fensters angezeigt.

Ist die Option „Diese Objekte auf dem Schreibtisch anzeigen“ für Netzwerk-Datenträger („Verbundene Server“) in den Finder-Voreinstellungen (Abb. 3.32) aktiviert, wird ein entsprechendes Icon ebenfalls auf dem Schreibtisch des Benutzers eingeblendet (Abb. 3.33). Die Verbindung kann in diesem Fall zusätzlich durch Ziehen des Netzwerk-Datenträgers auf das Papierkorb-Symbol im Dock oder durch die Auswahl des Befehls „Auswerfen...“ aus dem Kontext-Menü getrennt werden.

Diese Einblendung auf dem Schreibtisch ist allerdings nur ein Darstellungstrick des Finders und hat keine Entsprechung im Dateisystem. Insbesondere gibt es keinen Link o.Ä. im Verzeichnis `~/Desktop` des Benutzers, der auf das montierte Dateisystem zeigen würde.

Um das Dateisystem statt in `/Volumes` an irgendeiner anderen Stelle des Dateisystems einzublenden, muss man statt des Finders einen der oben gelisteten mount-Befehle verwenden. Neben der Spezifikation des zu verwendenden Servers und des dort jeweils freigegebenen Verzeichnisses kann bzw. muss man bei diesen Befehlen immer auch ein Verzeichnis angeben, an dessen Stelle das Dateisystem des jeweiligen Netzwerk-Datenträgers montiert werden soll.

3.5.1 Mit AFP-Servern verbinden

Apple Filing Protocol (AFP) ist ein Anwendungsprotokoll, welches heutzutage sowohl in TCP/IP- (AFP über TCP) als auch in AppleTalk-basierten Netzwerken (AFP über ASP) verwendet werden kann. Die Entwicklung von AFP begann bereits 1984 – damit ist AFP fast genauso alt wie der Ur-Macintosh. Zu den wesentlichen Entwicklungszielen gehörte neben der Unterstützung

von Macintosh-Besonderheiten[19] auch die Kompatibilität mit Unix- und MS-DOS-Systemen. Um diese Kompatibilität zu gewährleisten unterstützt AFP u. a. bis zu drei Namen pro Datei (Unicode, Macintosh und MS-DOS) sowie Unix-Zugriffsrechte.

Die ersten Versionen von AFP stützten sich ausschließlich auf die AppleTalk-Protokollfamilie – genauer gesagt auf das AppleTalk Session Protocol, kurz ASP. Erst mit AFP 2.2 (implementiert zum ersten Mal in AppleShare IP 5.0) wurde AFP von AppleTalk unabhängig. Apple spezifizierte mit dem Data Stream Interface (kurz DSI) eine Zwischenschicht, welche die Nutzung von AFP über beliebige datenstromorientierte Protokolle erlaubte. AFP über TCP (Port 548) blieb freilich die Hauptanwendung von DSI.

AFP-URLs

Zusätzlich zu Benutzername und Passwort kann bei AFP-URLs auch die Authentifizierungsmethode, engl. User Authentication Method (kurz UAM) angegeben werden.

Die allgemeine Form lautet dann:

```
afp://benutzer;AUTH=uam:passwort@server-name/pfad
```

Normalerweise lässt man den AFP-Client die Authentifizierungsmethode selbstständig mit dem AFP-Server aushandeln. Die explizite Angabe verwendet man häufig, um ein freigegebenes Verzeichnis ohne weitere Rückfragen als Gast zu aktivieren, da dazu die Methode `NO USER AUTHENT` verwendet werden muss. Um beispielsweise das Verzeichnis `/Users` vom Server `server.example.com` mit Gast-Rechten (ohne Rückfrage) zu aktivieren, müsste man die folgende URL angeben[20]:

```
afp://;AUTH=NO%20USER%20AUTHENT@server.example.com/Users
```

Um Verbindungen zu älteren AFP-Servern, die sich ausschließlich über AppleTalk ansprechen lassen, aufzubauen, muss eine leicht abgewandelte Form der URL verwendet werden:

```
afp:/at/benutzer;AUTH=uam:passwort@server-name:zone/pfad
```

Die Angabe der AppleTalk-Zone ist optional. Wird sie weggelassen, sucht der AFP-Client nach dem angegebenen Servernamen (hier natürlich ein AppleTalk-NBP-Name und kein TCP-DNS-Name) in der AppleTalk Zone, zu der der Rechner des Benutzers gehört. Ebenso können wie bei AFP über TCP-URLs auch der Benutzername, die Authentifizierungsmethode sowie das Passwort weggelassen werden.

[19] Dazu gehörten damals insbesondere Dateien mit einem Daten- und Ressourcen-Zweig und die Desktop-Datenbank des klassischen Finders.

[20] Leerzeichen und andere Sonderzeichen sollten in AFP-URLs in der üblichen Form kodiert werden. Obwohl der Finder auch URLs mit Leerzeichen akzeptiert, sollte man im Allgemeinen in allen URLs wie hier statt den Leerzeichen die Kodierung %20 verwenden.

Um das Verzeichnis `/Users` von einem Server mit dem AppleTalk-Namen `Server` mit Gast-Rechten zu aktivieren, müsste man demnach die folgende URL angeben:

```
afp:/at/;AUTH=NO%20USER%20AUTHENT@Server/Users
```

Verbindungsoptionen

Im Normalfall, wenn also in der AFP-URL weder Benutzername noch Passwort angegeben worden sind, erscheint nach einem Klick auf die Taste „Verbinden“ ein Authentifizierungsdialog, der zur Eingabe eines Benutzernamens und eines Passwortes auffordert. Falls der AFP-Server anonymen Zugriff erlaubt, kann man sich alternativ für die Anmeldung als „Gast“ entscheiden (Abb. 3.34). Handelt es sich beim AFP-Server um einen Mac-OS-X-Rechner, kann als Benutzername sowohl der vollständige Name (z. B. „Rafael Kobylinski“) als auch der Kurzname bzw. das Login (z. B. `rkk`) angegeben werden.

Über eine Checkbox kann die verschlüsselte Speicherung des angegebenen Passworts in der benutzereigenen Keychain (`~/Library/Keychains/login.keychain`) veranlasst werden. Bei späteren Verbindungsversuchen kann der Finder dieses Passwort wieder auslesen und den Authentifizierungsdialog gleich mit dem richtigen Passwort vorbelegen.

Der Authentifizerungsdialog verfügt links unten über ein Aktionsmenü mit zwei Einträgen. Der Menüeintrag „Kennwort ändern...“ ändert das Passwort des Benutzers auf dem entfernten Rechner. Der Menüeintrag „Optionen“ öffnet einen weiteren Dialog, in dem man einige Verbindungsparameter von AFP beeinflussen kann (Abb. 3.36).

Abb. 3.34. Nach Eingabe einer AFP-URL erscheint nach einem Klick auf die Taste „Verbinden“ ein Authentifizierungsdialog, der zur Eingabe eines Benutzernamens und eines Passwortes auffordert.

Abb. 3.35. Der Benutzer kann bequem sein Passwort auf dem entfernten Rechner ändern.

Abb. 3.36. Einige AFP-Verbindungsoptionen lassen sich direkt aus dem Authentifizierungsdialog heraus beeinflussen.

Die Option „Klartextübertragung des Kennworts erlauben“ dient der Kompatibilität zu älteren AFP-Servern, die keine sichereren Authentifizierungsmethoden unterstützen. Da die Authentifizierungsmethode während des Verbindungsaufbaus für den Anwender unsichtbar zwischen Client und Ser-

ver ausgehandelt wird, ist die Option „Warnung bei Klartextübertragung“ sinnvollerweise standardmäßig aktiviert.

Ist die Option „Sichere Verbindungen mithilfe SSH erlauben“ aktiv, versucht der Client nach der Authentifizierung einen SSH-Tunnel für die eigentliche Datenübertragung zwischen Client und Server aufzubauen. Auch hier erfolgt standardmäßig eine Warnung, falls der Server keine durch SSH abgesicherten AFP-Verbindungen unterstützt.

Der AFP-Client speichert seine Einstellungen in der (im Finder unsichtbaren) Datei `~/Library/Preferences/.GlobalPreferences.plist`[21]:

```
$ defaults read -g com.apple.AppleShareClientCore
{
    "afp_active_timeout" = 0;
    "afp_authtype_show" = 0;
    "afp_cleartext_allow" = 1;
    "afp_cleartext_warn" = 1;
    "afp_debug_level" = 6;
    "afp_debug_syslog" = 1;
    "afp_default_name" = "‘";
    "afp_idle_timeout" = 0;
    "afp_keychain_add" = 0;
    "afp_keychain_search" = 1;
    "afp_login_displayGreeting" = 1;
    "afp_mount_defaultFlags" = 0;
    "afp_prefs_version" = 1;
    "afp_reconnect_allow" = 1;
    "afp_reconnect_interval" = 10;
    "afp_reconnect_retries" = 12;
    "afp_ssh_allow" = 0;
    "afp_ssh_force" = 0;
    "afp_ssh_require" = 0;
    "afp_ssh_warn" = 0;
    "afp_use_default_name" = 0;
    "afp_use_short_name" = 0;
    "afp_voldlog_skipIfOnly" = 0;
    "afp_wan_quantum" = 0;
    "afp_wan_threshold" = 0;
}
```

Die Attribute, die man in dieser Datei findet, kann man natürlich auch manuell ändern (am einfachsten mit dem Property List Editor). Auf diese Weise lassen sich einige Dinge einstellen, für die der Finder derzeit noch keine Benutzerschnittstelle bietet. Sicherheitsbewusste Benutzer können z. B. mit Hilfe des Attributs `afp_ssh_require` die Nutzung von SSH erzwingen, so dass Verbindungsversuche zu Servern, die kein SSH unterstützen, abgebrochen werden.

Nach einer erfolgreichen Authentifizierung zeigt der AFP-Client eine Liste der auf dem ausgewählten AFP-Server freigegebenen Verzeichnisse an (Abb. 3.37). Der Benutzer kann eine oder mehrere dieser Verzeichnisse auswählen und als Netzwerk-Datenträger aktivieren.

[21] In Mac OS X 10.4 wird diese Datei im Binärformat gespeichert und muss konvertiert werden, um mit einem gewöhnlichen Texteditor betrachtet werden zu können. Der Property List Editor kann natürlich mit beiden Formaten umgehen.

Abb. 3.37. Nach einer erfolgreichen Authentifizierung zeigt der AFP-Client eine Liste der auf dem ausgewählten AFP-Server freigegebenen Verzeichnisse an.

Protokollierung von AFP-Verbindungen

Eine für die Fehlersuche interessante Option bieten die beiden Attribute `afp_debug_level` und `afp_debug_syslog`. Anhand dieser beiden Attribute entscheidet der AFP-Client nämlich, wie ausführlich und wohin etwaige AFP-Verbindungsversuche protokolliert werden sollen. Nach einer Erhöhung des Attributs `afp_debug_level` von 0 auf 7 sendet der AFP-Client eine Fülle von Statusmeldungen mit der Priorität `debug` an den Protokollierungsdienst `syslogd`.

Normalerweise werden Meldungen mit dieser Priorität nur dann angezeigt, wenn sie vom Betriebssystemkern stammen. Das lässt sich aber durch eine kleine Modifikation der Konfigurationsdatei `/etc/syslog.conf` ändern:

```
$ cat /etc/syslog.conf
*.err;kern.*;auth.notice;authpriv,remoteauth,install.none;mail.crit
        /dev/console
*.notice;authpriv,remoteauth,ftp,install.none;kern.debug;mail.crit
  /var/log/system.log

# Send messages normally sent to the console also to the serial port.
# To stop messages from being sent out the serial port, comment out this
  line.
#*.err;kern.*;auth.notice;authpriv,remoteauth.none;mail.crit
      /dev/tty.serial

# The authpriv log file should be restricted access; these
# messages shouldn't go to terminals or publically-readable
# files.
authpriv.*;remoteauth.crit                              /var/log/secure.log
```

```
lpr.info                                                /var/log/lpr.log
mail.*                                                  /var/log/mail.log
ftp.*                                                   /var/log/ftp.log
netinfo.err                                             /var/log/netinfo.log
install.*                                               /var/log/install.log
install.*                                               @127.0.0.1:32376
local0.*                                                /var/log/ipfw.log

*.emerg                                                   *
```

Um alle Meldungen der Priorität `debug` sichtbar zu machen, ersetzt man einfach in der zweiten Zeile dieser Datei den Text `kern.debug` durch `*.debug` und veranlasst eine Initialisierung von `syslogd`:

```
$ sudo killall -HUP syslogd
```

Anschließend lassen sich viele Details von AFP-Verbindungen in `/var/log/system.log` beobachten. Aufgrund der Informationsfülle sollte man die Protokollierung allerdings abschalten, wenn sie nicht mehr benötigt wird (Ausgabe stark gekürzt):

```
$ tail -f /var/log/system.log
...
... App Start: 1122222063 secs 021221 msecs
... Lookup Start: 1122222063 secs 163459 msecs
... DNSAddressResolver: DNS address Tiger-Client.local
... Load Modules Start: 1122222063 secs 166342 msecs
... Load Modules End: 1122222063 secs 183335 msecs
... Connect Start: 1122222063 secs 183454 msecs
... DNSAddressResolver:Resolve DNS address Tiger-Client.local
... DNSAddressResolver:Resolve found IPv6 address
...
... TUAMHandler::BuildUAMList noPrefs = 0 low level only = 0
... GetPascalCFString: Using the default string
... GetPascalCFString: Using the default string
... GetPascalCFString: Using the default string
... GetPascalCFString: Using the default string
... GetPascalCFString: Using the default string
... GetPascalCFString: Using the default string
... GetUAMsInDir: cannot find /~
... Looking at
... Looking at DHX2
...     Adding DHX2
... Looking at  DHCAST128
...     Adding  DHCAST128
... Looking at 2-Way Randnum exchange
... Looking at Cleartxt Passwrd
...     Adding Cleartxt Passwrd
... Looking at No User Authent
...     Adding No User Authent
...     DHX2
...             DHCAST128
...     Cleartxt Passwrd
...     No User Authent
... TUAMHandler::ChooseBestUAM
... Choosing DHX2
...
... TUAMHandlerCHI::Login DoLoginDialog returns = 0
```

```
... FetchVolumeList: 2 volumes total
... IsVolMounted: inMaxPath = 0 outMounPath = (null) mntFlags = 0000
... IsVolMounted: error = 2
... FetchVolumeList: Checking Access on Tiger Client  volflags = 00
...     FetchVolumeList: User has access
... IsVolMounted: inMaxPath = 0 outMounPath = (null) mntFlags = 0000
... IsVolMounted: error = 2
... FetchVolumeList: Checking Access on admin  volflags = 00
...     FetchVolumeList: User has access
... FetchVolumeList: 2 Volumes Available
... Vol Dlog Up: 1122222080 secs 893692 msecs
...
```

Um die Protokollierung wieder abzuschalten, muss in der zweiten Zeile von `/etc/syslog.conf` der Text `*debug` wieder durch `kern.debug` ersetzt, und `syslogd` erneut neu initialisiert werden.

Abbildung von Zugriffsrechten

Die effektiven Zugriffsrechte des Benutzers auf Objekte in einem aktivierten AFP-Dateisystem werden durch die verwendeten Benutzer-IDs und Kurznamen bestimmt.

Nach der Authentifizierung prüft der AFP-Client zunächst, ob die lokal (also auf dem Client-Rechner) verwendete Benutzer-ID mit der für die Authentifizierung am Server-Rechner verwendeten Benutzer-ID übereinstimmt. Ist das der Fall, prüft der AFP-Client zusätzlich, ob auch die beiden Kurznamen übereinstimmen.

Bei einer Übereinstimmung von Benutzer-IDs *und* Kurznamen nimmt der AFP-Client an, dass beide Rechner, der AFP-Client und AFP-Server, die gleichen Listen von Benutzern und Gruppen verwenden. Er übernimmt in diesem Fall die für die Verzeichnisse und Dateien auf dem AFP-Server definierten Zugriffsrechte eins zu eins.

In allen anderen Fällen findet eine so genannte Abbildung von Zugriffsrechten statt (engl. *privilege mapping*). Diese Abbildung ist erforderlich, weil die auf dem AFP-Server definierten Benutzer und Gruppen auf dem AFP-Client dann unbekannt sind. Die serverseitig definierten Eigentümer von Objekten können auf dem AFP-Client ebenso wenig dargestellt werden wie die ihnen zugewiesenen Gruppen.

Der AFP-Client bedient sich dafür dann auch eines Tricks. Er stellt alle Objekte so dar, als wäre der lokale Benutzer ihr Eigentümer und weist ihnen die Gruppe „unknown“ zu. Weist sich der lokale Benutzer `rkk` gegenüber dem Server als Benutzer `admin` aus, werden also alle Objekte als Eigentum von `rkk` dargestellt:

```
server:~ admin$ ls -l /Volumes/RAID/Projekte
total 0
drwxrwxr-x  8 admin  staff  272 23 Mar 19:36 Rot
drwxr-x---  2 root   staff   68 10 Aug 11:58 Gruen
drwx------  2 root   staff   68 10 Aug 11:54 Blau
```

```
...
client:~ rkk$ ls -l /Volumes/Projekte/
total 0
drwxr-xr-x  8 rkk  unknown  264 23 Mar 19:36 Rot
dr-x------  2 rkk  unknown  264 10 Aug 11:58 Gruen
d--x------  2 rkk  unknown  264 10 Aug 11:54 Blau
...
```

Die Rechte des Eigentümers auf dem Client entsprechen den Zugriffsrechten des für die Authentifizierung auf dem Server verwendeten Benutzers. Die Gruppenrechte auf dem Client (für die Gruppe „unknown“) sind einfach eine Kopie der Zugriffsrechte für den Rest der Welt (engl. others). Die Rechte für alle anderen Benutzer werden hingegen einfach vom Server übernommen.

Entsprechend hat der Benutzer `rkk` im obigen Beispiel Lese- und Schreibrechte für das Verzeichnis `Rot`, weil auf dem Server der Benutzer `admin` der Eigentümer dieses Verzeichnisses ist und damit die Eigentümerrechte übernommen werden konnten.

Anders beim Verzeichnis `Gruen`: hier hat der Benutzer `rkk` nur Lese-, aber keine Schreibrechte, da das Verzeichnis auf dem Server dem Benutzer `root` gehört. Der Grund ist hier die Gruppenzugehörigkeit des Benutzers `admin` zur Gruppe `staff`. In diesem Fall werden die serverseitigen Zugriffsrechte dieser Gruppe zu clientseitigen Zugriffsrechten des Benutzers `rkk`.

Beim Verzeichnis `Blau` ist schließlich auf dem Client-Rechner kein Zugriff möglich. Da serverseitig nur der Benutzer `root` als Eigentümer zugreifen darf, sind auf dem Client-Rechner weder Lese- noch Schreibzugriffe möglich.

Verbinden mit der Befehlszeile

Verbindungen zu AFP-Servern lassen sich auch über die Befehlszeile mit Hilfe des Befehls `mount_afp` recht leicht herstellen. Der Aufruf erfolgt mit mindestens zwei Parametern. Der erste Paramater ist einfach eine AFP-URL, der zweite der Pfad zu einem beliebigen, normalerweise leeren, Verzeichnis, welches als Anknüpfungspunkt für das entfernte Dateisystem dienen wird. Das Verzeichnis muss ggf. vorher erstellt werden:

```
$ mkdir mnt
$ mount_afp afp://server/Projekte mnt
mount_afp: the mount flags are 0000 the altflags are 0020
```

Wurde kein Benutzername und Passwort in der URL angegeben, erfolgt die Anmeldung ohne Authentifizierung, also mit Gast-Rechten. Über die Option `-i` lässt sich die Abfrage des Benutzernamens und des Passworts bei Bedarf jedoch aktivieren.

Nach dem Aufruf des `mount_afp`-Befehls erscheint der Inhalt des auf dem Server freigegebenen Verzeichnisses `Projekte` als Inhalt des lokalen Verzeichnisses `mnt`. Im Finder wird wie gehabt der Netzwerk-Datenträger `Projekte` symbolisch angezeigt. Letzteres lässt sich auf Wunsch mit Hilfe der Option `-o nobrowse` unterbinden. Um einen so aktivierten Datenträger ohne den

Finder wieder loszuwerden, kann der Befehl `umount` zusammen mit dem jeweiligen Anknüpfungspunkt verwendet werden:

```
$ umount mnt
```

War das als Anknüpfungspunkt verwendete Verzeichnis leer, wird es von `umount` automatisch gelöscht.

Die weiter oben beschriebene Protokollierung funktioniert auch mit dem Befehl `mount_afp` – allerdings nicht über `syslogd`. Man muss das Attribut `afp_debug_level` wie gehabt auf einen Wert von mindestens 7 setzen, sollte aber gleichzeitig das Attribut `afp_debug_syslog` mit dem Wert `false` versehen. Die Protokollierung erfolgt dann direkt über das Terminal:

```
$ mount_afp -o nobrowse afp://Tiger-Client.local/admin mnt
mount_afp: the mount flags are 100000 the altflags are 0020
AFPMountURL: No username & no UAMName forcing guest
SharedVolumeEnumerator::Init username = 0:???
DNSAddressResolver: DNS address Tiger-Client.local
DNSAddressResolver:Resolve DNS address Tiger-Client.local
DNSAddressResolver:Resolve found IPv6 address
...
```

3.5.2 Mit SMB-Servern verbinden

Server Message Block (SMB) ist ein ursprünglich ausschließlich für IBM-kompatible PCs entwickeltes Client-Server-Protokoll, welches den Zugriff auf entfernte Dateisysteme, Drucker und andere Ressourcen erlaubt. Seit 1996 verwendet Microsoft für seine Implementierung des Protokolls den Namen *Common Internet File System* (kurz CIFS).

SMB/CIFS stützt sich auf eine 1983 von Sytec und IBM entwickelte Programmbibliothek namens *Network Basic Input/Output System* (kurz NetBIOS). NetBIOS war die Programmierschnittstelle zu IBMs proprietärem Netzwerksystem „PC Network“, welches auf eine völlig eigenständige Hardware aufsetzte. Als Vernetzungstechnik für kleinere LANs unterstützte „PC Network“ maximal 72 Geräte, ohne Erweiterungsmöglichkeiten durch Router vorzusehen.

NetBIOS wurde später als *NetBIOS Extended User Interface* (kurz NetBEUI) weiterentwickelt und an Ethernet- sowie TokenRing-Netzwerksysteme angepasst. Um weiter entfernte Ressourcen erreichen zu können, ohne bestehende Software umschreiben zu müssen, wurde NetBIOS schließlich einfach in andere Protokolle wie IPX/SPX[22] und TCP/IP verpackt.

Heutzutage gebräuchlich und im Zusammenhang mit Mac OS X relevant ist vor allen Dingen die in TCP/IP verpackte Variante. *NetBIOS over TCP*, kurz NBT (manchmal auch NetBT), wurde 1987 in den RFCs 1001 und 1002 von der IETF standardisiert und besteht aus drei Komponenten:

[22] *Internet Packet Exchange/Sequenced Packet Exchange* – eine von Novell entwickelte, in den achtziger und neunziger Jahren recht verbreitete Protokollfamilie.

- einem Namensdienst, der die Abbildung von NetBIOS-Namen auf IP-Adressen (und umgekehrt) vornimmt;
- einer Implementierung von NetBIOS-Datagrammen auf der Basis von UDP (Port 138);
- sowie einer Implementierung von NetBIOS-Sessions auf der Basis von TCP (Port 139).

Auf diese Weise simuliert NBT ein lokales NetBIOS-Netzwerk (also quasi ein „PC Network“) mit Hilfe von TCP/IP. In seiner verbreitetsten Form nutzt SMB dieses simulierte Netzwerk, um den Zugriff auf entfernte Dateien und andere Ressourcen zu realisieren. Erst in der jüngsten Vergangenheit erschienen mit Windows 2000 und Windows XP SMB Implementierungen, welche direkt TCP/IP (TCP/UDP Port 445) ohne den Umweg über NBT verwenden.

Apple liefert mit Mac OS X 10.4.2 die Open-Source-Software Samba in der Version 3.0.10 aus:

```
$ smbd -V
Version 3.0.10
```

Samba ist eine freie und von Microsoft unabhängige Implementierung von SMB und erlaubt neben dem Zugriff auf mittels SMB freigegebene Ressourcen u. a. auch den Betrieb eines SMB-Servers. Samba lässt sich auf den meisten Unix-basierten Systemen – insbesondere auch auf Linux – problemlos installieren bzw. wird häufig, wie bei Mac OS X, bereits zusammen mit dem System ausgeliefert.

NetBIOS-Namen

Alle Hosts, die NBT beherrschen, haben außer einer IP-Adresse und einem DNS-Namen zusätzlich auch einen eindeutigen NetBIOS-Namen. Dieser NetBIOS-Name wird in der Regel bei der Installation festgelegt und kann vom Anwender jederzeit geändert werden. Um sicherzustellen, dass keine zwei Rechner den gleichen Namen verwenden, muss der NetBIOS-Name im Netzwerk „registriert“ werden.

Im einfachsten Fall erfolgt dies mit Hilfe eines lokalen Broadcasts des gewünschten NetBIOS-Namens beim Systemstart (bzw. beim Start des NBT-Subsystems). Wenn keine Einwände kommen, darf der Name verwendet werden. Sollte ein anderer Host den gleichen Namen bereits verwenden, kann er diesen Namen „verteidigen“, indem er den Broadcast mit einer entsprechenden negativen Antwort quittiert.

Da die meisten Router keine Broadcasts weiterleiten, ist diese Lösung für größere Netzwerke bzw. in einem Netzwerkverbund nicht praktikabel. Für diese Fälle sieht NBT den Betrieb eines *NetBIOS Name Servers* (kurz NBNS) vor. Falls dem startenden Host die IP-Adresse eines solchen Servers bekannt

Tabelle 3.1. Die NetBIOS-Knotentypen

Knotentyp	Verfahren
b-Knoten	verwendet ausschließlich das Broadcast-Verfahren
p-Knoten	verwendet ausschließlich den NBNS-Host (engl. unicast bzw. point-to-point)
m-Knoten	verwendet zuerst das Broadcast-Verfahren und dann, falls erforderlich, den NBNS-Host (engl. mixed)
h-Knoten	verwendet zuerst den NBNS-Host, und das Broadcast-Verfahren nur wenn der NBNS-Host nicht antwortet

ist, sendet er anstelle eines Broadcast eine an diesen Server gerichtete (unicast) Registrierungsanfrage. Der NBNS teilt dem Absender daraufhin mit, ob der gewünschte Name bereits vergeben ist oder nicht. *Windows Internet Name Service* (kurz WINS) ist die Implementierung eines NBNS von Microsoft und wird häufig als Synonym für NBNS verwendet.

Beide Verfahren kommen auch bei der Auflösung von NetBIOS-Namen zum Tragen. Um die zu einem NetBIOS-Namen passende IP-Adresse zu finden, kann ein NBT-Host eine Anfrage entweder per Broadcast-Verfahren an alle Hosts im Subnetz oder per Unicast-Verfahren an den designierten NBNS-Host schicken.

Jedem NBT-Host wird ein so genannter *Knotentyp* (engl. node type) zugewiesen, je nachdem, welches der beiden Verfahren bzw. in welcher Reihenfolge er die beiden Verfahren für die Registrierung des eigenen Namens und die Auflösung von fremden Namen verwendet (Tabelle 3.1).

NetBIOS-Namen sind in ihrer Länge auf 16 Bytes beschränkt. Davon werden die ersten 15 für den Host-Namen und das letzte Byte – für den Anwender unsichtbar – als Kennzeichen für einen Diensttyp verwendet (z. B. steht 32 für einen Fileserver). Für die Benennung stehen alle Buchstaben und Ziffern sowie einige Sonderzeichen (`! # $ % ' ( ) - . @ ^ _ { } ~`) zur Verfügung. Die Groß- und Kleinschreibung wird nicht beachtet: Jeder Name wird in eine Darstellung aus Großbuchstaben konvertiert, bevor er über ein Netzwerk übertragen wird. Häufig werden NetBIOS-Namen daher von Haus aus groß geschrieben[23].

Verbindungen mit dem Finder

Um im Netzwerk freigegebene Verzeichnisse in DOS und später Windows ansprechen zu können, hat Microsoft die so genannte *Universal Naming Convention* (kurz UNC) entwickelt. Ein UNC-Name hat eine gewisse Ähnlichkeit mit einer URL, wobei jedoch die Protokollkennzeichnung fehlt (es wird SMB

[23] Diese Großschreibung erleichtert auch die Unterscheidung zwischen NetBIOS- und DNS-Namen, da letztere üblicherweise klein geschrieben werden.

angenommen) und statt „normalen“ Schrägstrichen umgekehrte Schrägstriche verwendet werden:

```
\\server-name\pfad
```

Der „Verbinden mit Server…“-Dialog des Mac-OS-X-Finders verlangt hingegen nach einer SMB-URL. Als Server-Name kann dabei sowohl der NetBIOS-Name, der DNS-Name oder die IP-Adresse des Servers angegeben werden. Die folgenden Angaben sind unter der Voraussetzung, dass WINSERVER, 192.168.0.1 und server.meine-firma.de den gleichen Host bezeichnen, gleichwertig:

```
smb://WINSERVER/rkk

smb://192.168.0.1/rkk

smb://server.meine-firma.de/rkk
```

Die Pfadangabe kann weggelassen werden. Im Authentifizierungsdialog kann man die Arbeitsgruppe (oder die NT-Domäne), den Namen und das Passwort eingeben. Als einzige Option kann man das Passwort in der Keychain (verschlüsselt) speichern lassen, um es später nicht noch einmal eingeben zu müssen (Abb. 3.38).

Fehlte die Pfadangabe, erscheint nach erfolgreicher Authentifizierung ein Dialog, mit dessen Hilfe sich eines der vom ausgewählten Server freigegebenen Verzeichnisse auswählen lässt (Abb. 3.39). Dabei hat der Benutzer über die

Abb. 3.38. Im Authentifizierungsdialog kann man die Arbeitsgruppe (oder die NT-Domäne), den Namen und das Passwort eingeben.

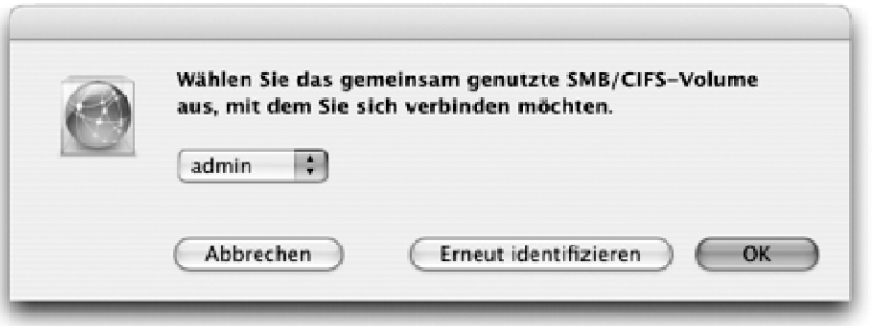


Abb. 3.39. Nach einer erfolgreichen Authentifizierung zeigt der SMB-Client eine Liste der auf dem ausgewählten SMB-Server freigegebenen Verzeichnisse an.

„Erneut Identifizieren"-Taste die Option, sich gegenüber dem SMB-Server auszuweisen, ohne ein freigegebenes Verzeichnis zu aktivieren. Da ohne Authentifizierung nur Verzeichnisse mit freigeschaltetem Gastzugriff angezeigt werden, werden auf diese Weise unter Umständen weitere freigegebene Ordner sichtbar.

Verbinden mit mount_smbfs

Mit dem Befehl `mount_smbfs` lassen sich Verbindungen zu SMB-Servern auch mit Hilfe der Befehlszeile herstellen. Der Befehl `mount_smbfs` erwartet den Pfad zu einem freigegebenen Verzeichnis in einer an UNC angelehnten Form (mit „normalen" Schrägstrichen) sowie ein lokales Verzeichnis als Anknüpfungspunkt. Ähnlich wie bei URLs dürfen Name und Passwort auch innerhalb der UNC angegeben werden, z. B.:

```
$ mount_smbfs //rkk:geheim@WINSERVER/rkk mnt
$ umount mnt
```

Dem NetBIOS-Namen darf der Name der Arbeitsgruppe vorangestellt werden. Er darf auch hier durch den DNS-Namen oder die IP-Adresse des Servers ersetzt werden. Anders als bei `mount_afp` wird beim Trennen der Verbindung das als Aktivierungspunkt verwendete Verzeichnis, auch wenn es leer ist, nicht gelöscht.

Wird kein Benutzername in der UNC angegeben, verwendet `mount_smbfs` einfach den Kurznamen des aktiven Benutzers und fragt nach einem Passwort.

```
$ mount_smbfs //WORKGROUP;WINSERVER/Projekte mnt
Password:
```

Ein anderer Benutzername kann mit Hilfe der Option `-U` angegeben werden. Wird diese Option ohne einen Benutzernamen oder mit einem Benutzernamen, der auf dem Server unbekannt ist, angegeben, erfolgt die Anmeldung als Gast.

Verbinden mit smbclient

Der Befehl `smbclient` kann als Werkzeug zur Übertragung von Dateien zwischen einem Mac-OS-X-Rechner und einem SMB-Server verwendet werden. Ähnlich wie im Finder lassen sich auch mit `smbclient` alle freigegebenen Verzeichnisse bzw. Freigaben eines Servers anzeigen:

```
$ smbclient -N -L //TIGER-SERVER
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (Tiger Server)
        ADMIN$          IPC       IPC Service (Tiger Server)
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.10]

        Server               Comment
        ---------            -------
        TIGER-CLIENT         Tiger Client
        TIGER-SERVER         Tiger Server

        Workgroup            Master
        ---------            -------
        WORKGROUP            TIGER-SERVER
```

Da sich die Liste der Freigaben auch ohne Authentifizierung anzeigen lässt, können wir auf die Angabe eines Benutzernamens und eines Passworts verzichten. Mit der Option `-N` schalten wir dann auch die ansonsten obligatorische Frage nach dem Passwort aus.

Anders als andere SMB-Clients zeigt `smbclient` auch versteckte Freigaben (das sind einfach alle Freigaben, deren Namen mit einem `$` enden): `IPC$` und `ADMIN$` im obigen Beispiel. Die Freigabe `IPC$` (engl. Inter-Process Communication) wird als Speicherort für spezielle Dateien (Named Pipes und Mailslots) verwendet, die für bestimmte Arten von protokollinterner Kommunikation von Bedeutung sind. Die Freigabe `ADMIN$` ist auf Windows-Rechnern nichts anderes als der Windows-„Systemordner“, also z. B. `C:\WINNT`, und kann u. a. für Fernwartung verwendet werden[24].

Der Samba-Server legt diese Freigabe ebenfalls automatisch an, allerdings nur um Kompatibilität mit *Advanced Server for Unix* (kurz ASU) von Hewlett-Packard zu gewährleisten. Die Samba-Variante enthält auch keine „echten“, sondern ähnlich wie `IPC$` ausschließlich IPC-Dateien, so dass sie im Vergleich zur `ADMIN$`-Freigabe unter Windows kaum ein Sicherheitsrisiko darstellt.

Kennt man den Namen eines freigegebenen Verzeichnisses, kann man mit `smbclient` eine Verbindung zu diesem Verzeichnis aufnehmen. Bei der Ver-

[24] Windows-Rechner legen zusätzlich noch versteckte Freigaben für alle angeschlossenen Laufwerke an, also z. B. C$ für das Festplattenlaufwerk C: und D$ für das Festplattenlaufwerk D:.

bindungsaufnahme kann man ähnlich wie bei `mount_smbfs` den Benutzernamen entweder explizit mit Hilfe der Option `-U` angeben, oder den auf dem Client-Rechner verwendeten Kurznamen implizit übernehmen. In beiden Fällen wird man zur Eingabe eines Passworts zur Authentifizierung aufgefordert. Nach einer erfolgreichen Authentifizierung steht einem eine befehlszeilenorientierte Umgebung mit zahlreichen Befehlen zur Verfügung:

```
$ smbclient -U admin //TIGER-SERVER/projekte
Password:
Domain=[TIGER-SERVER] OS=[Unix] Server=[Samba 3.0.10]
smb: \> help
?              altname       archive       blocksize     cancel
case_sensitive cd            chmod         chown         del
dir            du            exit          get           hardlink
help           history       lcd           link          lowercase
ls             mask          md            mget          mkdir
more           mput          newer         open          print
printmode      prompt        put           pwd           q
queue          quit          rd            recurse       reget
rename         reput         rm            rmdir         setmode
stat           symlink       tar           tarmode       translate
vuid           logon         !
smb: \>
```

Mit dem Befehl `help` (oder der Kurzform `?`) lassen sich kurze Hilfetexte zu den einzelnen Befehlen anzeigen:

```
smb: \> help ls
HELP ls:
        <mask> list the contents of the current directory

smb: \> ls
  .                   D        0  Fri Aug 13 17:38:58 2004
  ..                  D        0  Thu Jul 15 16:09:08 2004
  .DS_Store           H     6148  Tue Aug 10 12:52:19 2004
  Blau                D        0  Wed Aug 11 13:15:12 2004
  Gruen               D        0  Wed Aug 11 13:14:51 2004
  Rot                 D        0  Tue Mar 23 19:36:28 2004

        38159 blocks of size 4194304. 10148 blocks available
smb: \>
```

Einfache Datenübertragung lässt sich mit den Befehlen `get` (Datei vom Server auf den Client kopieren) und `put` (Datei vom Client auf den Server kopieren) bewerkstelligen. Mehrere Dateien gleichzeitig lassen sich mit den Befehlen `mget` und `mput` kopieren:

```
smb: \> cd Blau
smb: \Blau\> ls
  .                   D        0  Fri Aug 13 17:42:37 2004
  ..                  D        0  Fri Aug 13 17:38:58 2004
  Dunkelblau          D        0  Fri Aug 13 17:42:37 2004
  Hellblau            D        0  Fri Aug 13 17:42:34 2004

        38159 blocks of size 4194304. 10148 blocks available
smb: \Blau\> mget *
```

```
Get file Dunkelblau? y
Get file Hellblau? y
smb: \Blau\>
```

Die Rückfragen lassen sich mit dem Befehl `prompt` abschalten. Mit `mget` und `mput` lassen sich dann sehr schnell auch komplette Verzeichnisbäume kopieren – allerdings muss dazu vorher noch der Rekursivmodus aktiviert werden:

```
smb: \Blau\> cd ..
smb: \> recurse
smb: \> prompt
smb: \> mget Blau
smb: \>
```

Mit `quit` oder `exit` lässt sich `smbclient` beenden:

```
smb: \> exit
$
```

3.5.3 Mit NFS-Servern verbinden

Network File System (kurz NFS) ist ein von Sun Microsystems entwickeltes Protokoll zur gemeinsamen Nutzung von Dateisystemen in Unix-Netzwerken. Solche Netzwerke wurden in der Vergangenheit meistens zentral administriert, so dass die Benutzer sehr häufig nur sehr eingeschränkte Zugriffsrechte besaßen. Gleichzeitig war (und ist) die Nutzung eines Unix-Rechners durch mehrere Benutzer gleichzeitig, anders als bei klassischen Macs und IBM-kompatiblen PCs, in Unix-Netzwerken technisch mit Hilfe von Werkzeugen wie `rlogin`, `telnet` oder `ssh` einfach möglich und durchaus üblich.

Diesen Gegebenheiten entsprechend wurde NFS für den Einsatz in Rechnernetzen mit „vertrauenswürdigen“ Hosts und abhörsicheren Netzwerkverbindungen zwischen den Hosts geschaffen. In solchen Netzwerken verfügen ausschließlich vertrauenswürdige Personen über Administrationsrechte (also insbesondere das Root-Passwort) für die angeschlossenen Rechner, während alle normalen Benutzer nicht-privilegierte Kennungen verwenden. Gleichzeitig ist durch technische und organisatorische Maßnahmen sichergestellt, dass ausschließlich diese vertrauenswürdigen Personen zusätzliche Rechner an das Netzwerk anschließen können.

Da der NFS-Server die NFS-Clients als vertrauenswürdig betrachtet, verlangt er von einem NFS-Client konsequenterweise keinerlei Authentifizierung und verlässt sich stattdessen vollkommen auf die Korrektheit der übermittelten Benutzernummer (UID), die er für die Authorisierung der Dateizugriffe verwendet.

Der Benutzer von NFS muss sich also nicht durch ein Passwort legitimieren, um zu beweisen, dass er tatsächlich derjenige ist, der er vorgibt zu sein. Der NFS-Server nimmt stattdessen einfach an, dass die Authentifizierung bereits bei der Anmeldung des Benutzers am Client-Rechner erfolgt ist.

Entsprechend darf der Benutzer natürlich nur auf diejenigen Verzeichnisse und Dateien zugreifen, für die er über die entsprechenden Zugriffsrechte verfügt, also für die er serverseitig authorisiert ist. Diese Authorisierung erfolgt in der Regel ganz normal über die Vergabe von Zugriffsrechten auf dem exportierten Dateisystem des NFS-Servers.

Da die Identifikation der NFS-Benutzer anhand ihrer Benutzernummern erfolgt, setzt NFS eine zwischen Client- und Server-Rechner abgeglichene Benutzerdatenbank voraus. Als Minimalvoraussetzung muss man jedem Benutzer eine netzwerkweit eindeutige Benutzernummer zuordnen. Obwohl es ansonsten für die Nutzung von NFS egal ist, ob man den notwendigen Abgleich manuell, durch Kopieren der entsprechenden Konfigurationsdateien oder durch den Einsatz eines Verzeichnisdienstes bewerkstelligt, wird man in der Praxis NFS nahezu ausschließlich zusammen mit einem Verzeichnisdienst verwenden wollen, um den Abgleich zu automatisieren.

NFS-Freigaben werden häufig beim Systemstart aktiviert und stehen allen Benutzern des Client-Rechners zur Verfügung. Da die Zugriffsrechte der Freigabe bei der Aktivierung nicht auf die Zugriffsrechte des Aktivierenden abgebildet werden, können auf dem Client-Rechner auch mehrere Benutzer gleichzeitig auf die Freigabe zugreifen, ohne dass es zu Problemen mit Zugriffsrechten kommen würde.

Verbinden mit dem Finder

Um mit Hilfe des Finder-Dialogs „Verbinden mit Server“ eine Verbindung zu einem NFS-Server aufbauen zu können, muss man neben dem DNS-Namen bzw. der IP-Adresse des Servers auch den vollständigen Pfad zur gewünschten Freigabe kennen.

Im Gegensatz zu AFP- oder SMB-Freigaben verfügen NFS-Freigaben dabei nicht über einen eigenständigen Namen, sondern werden einfach mit dem lokalen Pfad, über den sie auf dem Server lokal zu erreichen sind, identifiziert.

Der Befehl `showmount` erlaubt die Auflistung aller auf einem NFS-Server definierten Freigaben:

```
$ showmount -e server
Exports list on server:
/Volumes/RAID/Projekte             192.168.0.0
/Volumes/RAID/Applications          Everyone
```

Die hier im Beispiel angezeigten Freigaben lassen sich im Finder entsprechend mit den folgenden URLs aktivieren (Abb. 3.40):

```
nfs://server/Volumes/RAID/Projekte
nfs://server/Volumes/RAID/Applications
```

Verbinden mit mount_nfs

Mit `mount_nfs` kann man Verbindungen zu einem NFS-Server mit der Befehlszeile herstellen und mit `umount` wieder trennen.

Abb. 3.40. Um NFS-Freigaben zu aktivieren, muss man neben dem DNS-Namen bzw. der IP-Adresse des Servers auch deren vollständigen Pfad kennen.

Der Befehl erwartet als ersten Parameter den DNS-Namen bzw. die IP-Adresse des Servers, einen Doppelpunkt, sowie den vollständigen Pfad der NFS-Freigabe. Als zweiten Parameter muss man ein lokales Verzeichnis als Anknüpfungspunkt angeben:

```
$ mount_nfs server:/Volumes/RAID/Projekte mnt
$ umount
```

3.5.4 Mit FTP-Servern verbinden

File Transfer Protocol (kurz FTP) ist ein auf Unix-Systemen sehr verbreitetes Standard-Protokoll zur Übertragung von Dateien in einem Netzwerk. Im Gegensatz zu Protokollen wie AFP, SMB oder NFS ist mit FTP der wahlfreie Zugriff auf Dateien nicht möglich. Um Teile einer Datei zu öffnen, muss die komplette Datei zuerst vom Server-Rechner auf den Client-Rechner kopiert werden und kann erst dann geöffnet werden.

FTP ist ein verhältnismäßig einfaches Protokoll, welches zwei TCP-Verbindungen verwendet. Die erste Verbindung dient dabei der Steuerung, die zweite der eigentlichen Datenübertragung. Ein FTP-Server erwartet Steuerungsverbindungen auf dem TCP-Port 21. Nachdem ein FTP-Client eine Steuerungsverbindung aufgebaut hat, muss er sich gegenüber dem Server mit Hilfe eines Benutzernamens und eines Passworts authentifizieren.

Für die eigentliche Datenübertragung wird eine zweite TCP-Verbindung aufgebaut. Je nach Übertragungsmodus wird diese zweite TCP-Verbindung entweder vom FTP-Server oder vom FTP-Client initiiert. Im *Aktivmodus* teilt der FTP-Client dem FTP-Server lediglich seine IP-Adresse und eine Port-Nummer mit. Der FTP-Server baut daraufhin von sich aus eine Verbindung zum FTP-Client auf. Im *Passivmodus* teilt der FTP-Server dem FTP-Client eine Port-Nummer mit und es liegt in der Verantwortung des FTP-Clients,

die zweite Verbindung herzustellen. Im Aktivmodus wird also die Steuerungsverbindung vom FTP-Client zum FTP-Server, und die Datenverbindung in der umgekehrten Richtung, vom FTP-Server zum FTP-Client, aufgebaut. Im Passivmodus gehen beide Verbindungen vom FTP-Client aus.

Aufgrund dieser Besonderheit ist FTP mit automatischer Adressübersetzung (NAT) nur unter bestimmten Bedingungen kompatibel. Der Betrieb eines FTP-Clients auf einem Rechner mit einer privaten IP-Adresse, der Verbindungen in das öffentliche Internet nur über einen NAT-Router herstellen kann, ist im Normalfall nur im Passivmodus möglich. Umgekehrt ist der Betrieb eines FTP-Servers hinter einem NAT-Router – mit Weiterleitung des TCP-Ports 21 – normalerweise nur im Aktivmodus erfolgreich. Diese Einschränkungen lassen sich nur durch den Einsatz eines NAT-Routers umgehen, der FTP-Steuerungsverbindungen erkennen kann und entsprechend modifiziert.

Eine weitere Schwäche von FTP ist die Übertragung des für die Authentifizierung benutzten Benutzernamens und des dazugehörigen Passworts im Klartext. Aus diesem Grunde wird FTP heutzutage vorrangig für die Bereitstellung von Dateien für den anonymen Zugriff (engl. anonymous ftp) verwendet.

Verbinden mit dem Finder

Obwohl FTP im Gegensatz zu AFP, SMB oder NFS keine wahlfreien Zugriffe unterstützt und über FTP freigegebene Verzeichnisse daher keine Netzwerklaufwerke im engeren Sinne darstellen, können im Finder auch FTP-Server als Netzwerkdatenträger aktiviert werden.

Nachdem man eine FTP-URL im Dialog mit „Mit Server verbinden“ eingegeben hat, wird man vom Finder zur Eingabe eines Benutzernamens und eines Passworts aufgefordert (Abb. 3.41). Da FTP nur ein freigegebenes Verzeichnis pro FTP-Server unterstützt, muss bei der Eingabe der URL kein Pfad mit angegeben werden. Gibt man trotzdem einen Pfad an, dann kann man an Stelle des freigegebenen Verzeichnisses eines seiner Unterverzeichnisse aktivieren:

```
ftp://server/
ftp://server/Music
```

Nach dem Aktivieren eines FTP-Verzeichnisses sollte man allerdings nicht vergessen, dass es sich um FTP und nicht etwa um eines der anderen Protokolle handelt. Obwohl es Mac OS X gelingt, die beim Öffnen von Dateien nötigen Kopiervorgänge vor dem Anwender gänzlich zu verstecken, kann es zu unnötigen Wartezeiten kommen, wenn man versucht, eine große Datei zu öffnen, nur um einige kleine Änderungen zu machen.

Beim einfachen Doppelklick auf eine Datei oder ein Programm bekommt man zur Sicherheit einen (bei Dateien etwas irreführenden) Hinweis zu sehen, der an die Tatsache erinnert, dass man mit einem FTP-Server arbeitet (Abb. 3.42).

Abb. 3.41. Obwohl FTP im Gegensatz zu AFP, SMB oder NFS keine wahlfreien Zugriffe unterstützt, können im Finder auch FTP-Server als ebensolche aktiviert werden.

Abb. 3.42. Bei einem FTP-Server ist Öffnen gleichbedeutend mit Kopieren, weswegen die Finder-Entwickler wohl einen (bei Dateien etwas irreführenden) Warnhinweis für angebracht hielten.

Ansonsten ist die FTP-Unterstützung des Finders recht rudimentär. Es ist ausschließlich lesender Zugriff möglich, und zwar auch dann, wenn der FTP-Server Schreibzugriff unterstützen würde. Immerhin lässt sich in Systemeinstellungen Netzwerk über eine Checkbox angeben, ob der eingebaute FTP-Client den Passivmodus verwenden soll oder nicht. Ebenso lässt sich dort ein FTP-Proxy angeben, falls kein direkter FTP-Zugriff erlaubt sein sollte (Abb. 3.43).

Abb. 3.43. In den Systemeinstellungen Netzwerk kann man über eine Checkbox angeben, ob der FTP-Client den Passivmodus verwenden soll oder nicht. Ebenso lässt sich hier ein FTP-Proxy angeben, falls kein direkter FTP-Zugriff erlaubt sein sollte.

Verbinden mit mount_ftp

Mit `mount_ftp` kann man im Verhalten dem Finder entsprechende Verbindungen zu FTP-Servern auch per Befehlszeile herstellen. Die Syntax von `mount_ftp` ähnelt dabei der Syntax von `mount_nfs`. Auch hier wird als erster Parameter der DNS-Name oder die IP-Adresse des Servers gefolgt von einem Doppelpunkt und einem Pfad erwartet. Als zweiter Parameter muss ebenfalls ein lokales Verzeichnis als Anknüpfungspunkt angegeben werden. Anders als bei `mount_nfs` ist hier der Pfad allerdings optional: wird er angegeben, dann wird an Stelle des per FTP freigegebenen Verzeichnisses das

entsprechende Unterverzeichnis aktiviert; fehlt der Pfad, dann wird einfach das designierte FTP-Verzeichnis des FTP-Servers aktiviert:

```
$ mount_ftp server mnt
$ umount mnt
$ mount_ftp server:/Music
$ umount mnt
```

Wie bei allen Befehlen der mount-Familie üblich, lassen sich auch Verbindungen zu FTP-Servern mit Hilfe des `umount`-Befehls wieder trennen.

Verbinden mit ftp

Neben der komfortablen, aber rudimentären FTP-Unterstützung des Finders steht in Mac OS X mit `ftp` natürlich ein vollständiger FTP-Client in der Kommandozeile zur Verfügung.

Der Benutzer kann `ftp` ohne Argumente aufrufen und die Verbindung zu einem bestimmten FTP-Server innerhalb des Programms aufbauen, oder alternativ einen Hostnamen als Parameter mitgeben.

Der Hostname kann dabei mit oder ohne Benutzername sowie mit oder ohne Pfad angegeben werden. Er darf entweder in der bei `mount_ftp` bzw. `mount_nfs` verwendeten Syntax oder als URL angegeben werden:

```
$ ftp server
$ ftp rkk@server
$ ftp rkk@server:/Music
$ ftp ftp://server
$ ftp ftp://rkk:geheim@server/Music
```

Eine typische FTP-Sitzung beginnt mit der Authentifizierung:

```
$ ftp server
Connected to server.example.de.
220----------------------------------------------------------------------
220-This is the "Banner" message for the Mac OS X Server's FTP server
    process.
220-
220-              FTP clients will receive this message immediately
220-              before being prompted for a name and password.
220-
220-PLEASE NOTE:
220-
220-       Some FTP clients may exhibit problems if you make this file too
           long.
220-
220----------------------------------------------------------------------
220-
220 server.example.de FTP server (Version:  Mac OS X Server 10.3.2 -
    +GSSAPI) ready.
Name (server:rkk):
331 Password required for rkk.
Password:
230----------------------------------------------------------------------
230-This is the "Welcome" message for the Mac OS X Server's FTP server
```

```
   process.
230-
230-FTP clients will receive this message right after a successful log in.
230-
230----------------------------------------------------------------------
230-
230 User rkk logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Um sich als Gast ohne ein Passwort anmelden zu können, muss man `anonymous` als Benutzernamen angeben. Sofern der FTP-Server Gastverbindungen unterstützt, wird er den Benutzer zur Eingabe seiner Email-Adresse o.Ä. als Passwort auffordern (Falls der FTP-Server hingegen keine Gastverbindungen unterstützt, wird er in der Regel daraufhin die Verbindung trennen).

Nach erfolgreicher Anmeldung stehen innerhalb des FTP-Clients zahlreiche Unterbefehle zur Verfügung, die sich mit `help` auflisten lassen. Zusätzlich kann man sich für die meisten von ihnen kurze Hilfetexte anzeigen lassen:

```
ftp> help
Commands may be abbreviated.  Commands are:

!            disconnect   lpage        nlist        rate         sndbuf
$            edit         lpwd         nmap         rcvbuf       status
account      epsv4        ls           ntrans       recv         struct
append       exit         macdef       open         reget        sunique
ascii        features     mdelete      page         remopts      system
bell         fget         mdir         passive      rename       tenex
binary       form         mget         pdir         reset        throttle
bye          ftp          mkdir        pls          restart      trace
case         gate         mls          pmlsd        rhelp        type
cd           get          mlsd         preserve     rmdir        umask
cdup         glob         mlst         progress     rstatus      unset
chmod        hash         mode         prompt       runique      usage
close        help         modtime      proxy        send         user
cr           idle         more         put          sendport     verbose
debug        image        mput         pwd          set          xferbuf
delete       lcd          msend        quit         site         ?
dir          less         newer        quote        size
ftp> help cd
cd              change remote working directory
ftp> help ls
ls              list contents of remote path
ftp>
```

Die Befehle `cd`, `pwd` und `ls` funktionieren ähnlich wie in einer Shell: Mit `cd` kann man das aktuelle (Arbeits-)Verzeichnis wechseln, mit `pwd` den Pfad zum aktuellen Verzeichnis anzeigen lassen und mit `ls` seinen Inhalt auflisten (Ausgabe leicht gekürzt):

```
ftp> pwd
257 "/" is current directory.
ftp> ls
227 Entering Passive Mode (192,168,0,2,180,5)
150 Opening ASCII mode data connection for directory listing.
```

```
total 56
drwxrwxr-x  1 root  admin  204 Aug 16 17:08 Projekte
...
226 Transfer complete.
ftp> cd Projekte
250 CWD command successful.
ftp> pwd
257 "/Projekte" is current directory.
ftp> ls
227 Entering Passive Mode (192,168,0,2,180,20)
150 Opening ASCII mode data connection for directory listing.
total 0
drwxr-xr-x  4 admin  staff  136 Aug 13 17:42 Blau
drwxr-xr-x  3 admin  staff  102 Aug 11 13:14 Gruen
drwxr-xr-x  8 admin  staff  272 Mar 23 19:36 Rot
226 Transfer complete.
ftp>
```

Wie man an den Ausgaben im Beispiel sieht, ist der Passivmodus voreingestellt[25]. Mit dem Befehl `passive` lässt sich aber jederzeit zwischen aktivem und passivem Modus umschalten:

```
ftp> ls
227 Entering Passive Mode (192,168,0,2,180,175)
150 Opening ASCII mode data connection for directory listing.
total 0
drwxr-xr-x  4 admin  staff  136 Aug 13 17:42 Blau
drwxr-xr-x  3 admin  staff  102 Aug 11 13:14 Gruen
drwxr-xr-x  8 admin  staff  272 Mar 23 19:36 Rot
226 Transfer complete.
ftp> passive
Passive mode: off; fallback to active mode: off.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for directory listing.
total 0
drwxr-xr-x  4 admin  staff  136 Aug 13 17:42 Blau
drwxr-xr-x  3 admin  staff  102 Aug 11 13:14 Gruen
drwxr-xr-x  8 admin  staff  272 Mar 23 19:36 Rot
226 Transfer complete.
ftp>
```

Um einzelne Dateien zu kopieren, kann man die Befehle `get` und `put` verwenden: mit `get` kopiert man eine Datei vom FTP-Server auf den FTP-Client und mit `put` kann man, sofern der FTP-Server Schreibzugriff zulässt, eine Datei vom FTP-Client auf den FTP-Server kopieren.

Um mehrere Dateien auf einmal, oder sogar komplette Verzeichnisse zu kopieren, kann man die Befehle `mget` und `mput` verwenden:

```
ftp> mget Blau
mget Blau/Dunkelblau [anpqy?]? y
200 PORT command successful.
150 Opening BINARY mode data connection for Blau/Dunkelblau
   (0 bytes).
226 Transfer complete.
```

[25] FTP verwendet die zusätzliche Datenverbindung nicht nur für die Übertragung von Dateien, sondern auch für die Übertragung der Inhaltsverzeichnisse.

```
mget Blau/Hellblau [anpqy?]? y
200 PORT command successful.
150 Opening BINARY mode data connection for Blau/Hellblau
    (0 bytes).
226 Transfer complete.
ftp> prompt
Interactive mode off.
ftp> mget Blau
local: Blau/Dunkelblau remote: Blau/Dunkelblau
200 PORT command successful.
150 Opening BINARY mode data connection for Blau/Dunkelblau
    (0 bytes).
226 Transfer complete.
local: Blau/Hellblau remote: Blau/Hellblau
200 PORT command successful.
150 Opening BINARY mode data connection for Blau/Hellblau
    (0 bytes).
226 Transfer complete.
ftp>
```

Wie man sieht, lassen sich die Rückfragen von `mget` und `mput` mit dem Befehl `prompt` deaktivieren.

Die Übertragung von Dateien mittels FTP findet normalerweise in einer von zwei Betriebsarten statt: binär (engl. binary) oder ASCII. Voreingestellt ist der Binärmodus – um in den ASCII-Modus zu schalten, kann man den Befehl `ascii`, und um in den Binärmodus zurückzuschalten, den Befehl `binary` verwenden.

Im Binärmodus werden die Dateien eins zu eins übertragen – das ist heutzutage die am häufigsten verwendete Betriebsart. Der ASCII-Modus dient hingegen ausschließlich zur Übertragung von Textdateien und soll für eine Übersetzung der übertragenen Zeichen in ein für die Zielplattform geeignetes Format sorgen. Eine Übersetzung kann notwendig werden, wenn Client und Server unterschiedliche Zeichentabellen, wie z. B. ASCII und EBCDIC[26], oder trotz gleicher Zeichentabelle unterschiedliche Zeichen für das Zeilenende verwenden. So ist auf dem Mac das Zeichen *Carriage Return* (kurz CR, ASCII-Wert 13), auf Unix-Systemen das Zeichen *Line Feed* (kurz CR, ASCII-Wert 10) und auf MS-DOS- und MS-Windows-Systemen die Zeichenkombination *CR/LF* als Markierung für das Zeilenende in Textdateien gebräuchlich[27].

Hat man die Dateiübertragung abgeschlossen, kann man die Verbindung zum FTP-Server mit dem Befehl `close` trennen. Eine neue Verbindung zu einem anderen (oder natürlich auch dem gleichen) FTP-Server lässt sich dann mit dem Befehl `open` initiieren. Möchte man hingegen die FTP-Sitzung beenden, dann kann man mit `quit` oder `bye` den FTP-Client verlassen:

```
...
ftp> quit
```

[26] Extended Binary Coded Decimals Interchange Code – eine auf IBM-Großrechnern gebräuchliche Art, Schriftzeichen als Zahlen zu kodieren.

[27] Unter Mac OS X findet man entsprechend der Entwicklungsgeschichte dieses Systems sowohl Textdateien, deren Zeilen mit CR enden, als auch welche, die mit LF enden.

```
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 47 bytes in 0 transfers.
221-Thank you for using the FTP service on server.example.de.
221 Goodbye.
```

3.5.5 Mit WebDAV-Servern verbinden

Das *Web-based Distributed Authoring and Versioning* (kurz WebDAV)-Protokoll ist eine Erweiterung des *Hypertext Transfer Protocols* (kurz HTTP/1.1) um Dateibearbeitungs- und Dateiverwaltungsfunktionen[28].

Mit Hilfe von HTTP kann man normalerweise nur lesend auf die von einem HTTP-Server bereitgestellten Dateien zugreifen. Da das *World Wide Web* (kurz WWW) die wichtigste Anwendung von HTTP darstellt, verwendet der Benutzer meistens einen Web-Browser (z. B. Apple Safari oder Microsoft Internet Explorer) als HTTP-Client. Der Web-Browser verwendet seinerseits HTTP, um die auf einem HTTP-Server (Web-Server) bereitgestellten Dateien auf den Client-Rechner zu kopieren. Dieser Kopiervorgang ist für den Benutzer nahezu unsichtbar, da die kopierten Dateien entweder nur im Arbeitsspeicher, oder in einem versteckten Bereich des Dateisystems zwischengespeichert werden. Bei den vom Web-Browser mittels HTTP kopierten Dateien handelt es sich in der Regel um Dokumente im HTML-Format (engl. Hypertext Markup Language), die von ihm sofort interpretiert und direkt auf dem Bildschirm dargestellt werden.

WebDAV ermöglicht in Ergänzung dazu Schreibzugriffe auf die von einem HTTP-Server bereitgestellten Dateien. In der Praxis wird es heute häufig von Web-Seiten-Gestaltern eingesetzt, um die bearbeiteten Dateien auf einem HTTP-Server zu publizieren (manche Programme, z. B. Adobe GoLive, unterstützen dazu WebDAV auch direkt). Im Prinzip ist WebDAV aber nicht auf solche Web-nahen Anwendungsfälle beschränkt: so nutzt z. B. Apple Computer WebDAV, um .Mac-Abonnenten den Zugriff auf die *iDisk* zu ermöglichen.

Mit Hilfe von WebDAV lassen sich die üblichen Dateioperationen, wie z. B. Datei kopieren, Datei anlegen, Datei löschen, Datei umbenennen, Verzeichnis anlegen u.Ä. durchführen. Konflikte, die beim gleichzeitigen Zugriff von mehreren Benutzern auf das gleiche Verzeichnis auftreten können, werden mit Hilfe von Sperren automatisch gelöst. Dabei unterscheidet man zwischen Klasse-2-WebDAV-Servern, die solche Sperren unterstützen, und einfacheren Klasse-1-WebDAV-Servern, die solche Sperren nicht unterstützen.

Da beim Lese- und beim Schreibzugriff, ähnlich wie bei FTP, immer die komplette Datei übertragen werden muss, ist WebDAV allerdings nicht unbedingt als Ersatz für spezialisierte Dateizugriffsprotokolle wie AFP, SMB oder NFS geeignet.

[28] Die ursprünglich geplante Versionsverwaltung für die gespeicherten Dokumente war in der ersten Spezifikation von WebDAV nicht enthalten und spielt derzeit in der Praxis noch keine Rolle.

Günstig ist jedoch in vielen Fällen die technische Nähe zu HTTP. Da WebDAV auf HTTP aufbaut, kann man auf WebDAV-Server meistens ohne durch Firewalls behindert zu werden zugreifen. Für sicheren Zugriff lassen sich WebDAV-Verbindungen analog zu HTTP-Verbindungen mit Hilfe von SSL (engl. Secure Sockets Layer) verschlüsseln.

Verbinden mit dem Finder

WebDAV-Server lassen sich wie alle anderen Server mit Hilfe des „Mit Server verbinden“-Dialogs aktivieren. Da WebDAV auf HTTP aufbaut, werden WebDAV-Server mit HTTP-URLs angesprochen:

```
http://server
http://192.168.0.2
```

Sofern auf dem WebDAV-Server die Zugriffskontrolle aktiviert ist, erscheint nach Eingabe der URL ein Authentifizierungsdialog, der zur Eingabe des Benutzernamens und des dazugehörigen Passworts auffordert (Abb. 3.44).

Das über WebDAV freigegebene Verzeichnis erscheint im Finder wie gewohnt als Netzwerkdatenträger. Sofern es sich beim Server um einen Klasse-2-WebDAV-Server handelt, sind sowohl Lese- als auch Schreibzugriffe auf das WebDAV-Verzeichnis erlaubt. Klasse-1-WebDAV-Server (die keine Sperren bei Mehrfachzugriff unterstützen) werden aus Sicherheitsgründen ausschließlich im Nur-Lese-Modus aktiviert, um die Beschädigung von Dateien beim parallelen Zugriff durch mehrere Benutzer gleichzeitig auszuschließen.

Abb. 3.44. WebDAV-Authentifizierung.

Verbinden mit mount_webdav

Alternativ kann man WebDAV-Verzeichnisse auch über den Befehl `mount_webdav` aktivieren. Als erster Parameter wird der Servername (ggf. mit Pfad), als zweiter Parameter ein lokales Verzeichnis als Anknüpfungspunkt erwartet:

```
$ mkdir mnt
$ mount_webdav server mnt
$ umount
```

3.5.6 Der Finder als Netzwerk-Browser

Um Verbindungen zu Servern mit Hilfe des „Verbinden mit Server...“-Dialogs aufbauen zu können, benötigt man, je nach Protokoll, die IP-Adresse, den DNS-Namen, den WINS-Namen oder den AppleTalk-Namen des Servers.

Um nach Servern in der Umgebung automatisch zu suchen, verfügt der Finder aber auch über eine integrierte Netzwerk-Browserfunktion. Der Netzwerk-Browser des Finders wird durch einen Ordner mit dem Namen „Netzwerk“ symbolisiert.

Der Inhalt dieses Ordners entspricht dem Inhalt des Verzeichnisses `/Network/` auf der Systempartition. Dieser Inhalt ist jedoch nicht statisch, sondern wird dynamisch vom Hintergrundprozess `automount` verwaltet. Dieser Prozess wird innerhalb des StartupItems NFS gestartet und ist für die Inhalte dieses Verzeichnisses zuständig.

Welche Inhalte in diesem Verzeichnis auftauchen, ist dabei einerseits von der jeweiligen Konfiguration und andererseits von der Netzwerkumgebung abhängig, in die man eingebunden ist.

Mit dem Programm *Verzeichnisdienste* kann man festlegen, welche Protokolle für die Suche nach aktiven Servern verwendet werden sollen. Zur Auswahl stehen AppleTalk[29], Bonjour, SMB und SLP (Abb. 3.45).

In Netzwerken mit nur einer AppleTalk-Zone tauchen die NBP-Namen von AppleTalk-Servern direkt im Netzwerk-Verzeichnis auf. Wird ein AppleTalk-Router betrieben und gibt es zwei oder mehr Zonen, dann werden diese als Unterverzeichnisse des Netzwerk-Verzeichnisses repräsentiert. Die AppleTalk-Server-Namen tauchen dann entsprechend ihrer Zonen-Zugehörigkeit in dem jeweiligen Unterverzeichnis auf.

In Mac OS X 10.3 wurde für Namen von Bonjour-, SMB- und SLP-Servern grundsätzlich mindestens ein Unterverzeichnis angelegt. Das Bonjour-Verzeichnis hieß grundsätzlich `Local`, während die Namen der SMB- und SLP-Verzeichnisse jeweils den Namen der NetBIOS-Arbeitsgruppe bzw. des SLP-Bereichs trugen.

In Mac OS X 10.4 werden solche Verzeichnisse offenbar erst dann angelegt, wenn das Netzwerk komplexer ist, also wenn beispielsweise mehrere Apple-

[29] Damit die Suche nach AppleTalk-Servern gelingt, muss natürlich das AppleTalk-Protokoll für mindestens eine Netzwerkschnittstelle aktiviert werden.

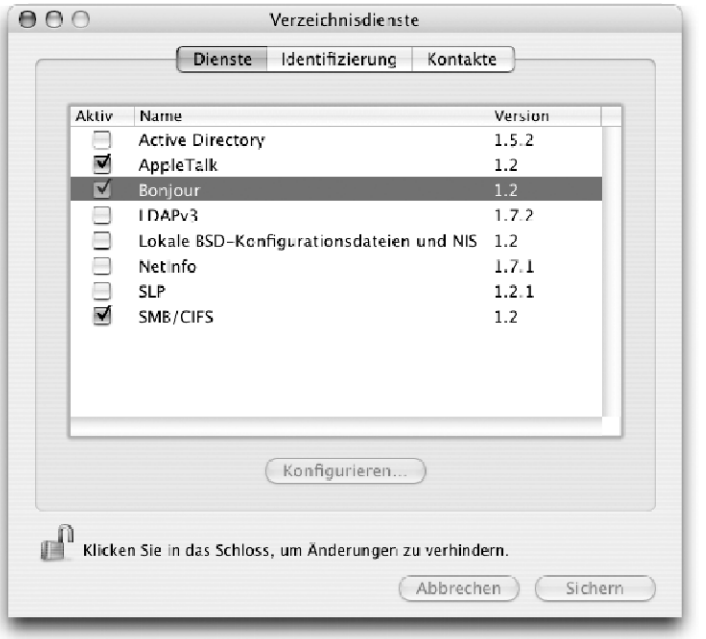


Abb. 3.45. Mit dem Programm *Verzeichnisdienste* kann man festlegen, welche Protokolle für die Suche nach aktiven Servern verwendet werden sollen.

Talk-Zonen vorliegen. In einfachen Fällen tauchen die Namen der gefundenen Server einfach innerhalb des Netzwerk-Verzeichnisses auf.

Um mit Hilfe des Netzwerk-Browsers eine Verbindung aufzubauen, muss man mit Hilfe des Finders zum jeweiligen Server-Namen navigieren und das Objekt „öffnen“. Ab da verläuft der Verbindungsaufbau abhängig vom verwendeten Protokoll genauso, als hätte man die Verbindung über den „Verbinden mit Server...“-Dialog hergestellt[30].

3.6 Verzeichnisse selbst freigeben

3.6.1 AFP-Server aktivieren

Jeder Mac-OS-X-Rechner kann die Rolle eines AFP-Servers übernehmen. In Mac OS X 10.4 kann man den AFP-Server in den Systemeinstellungen Sharing aktivieren. Um den AFP-Server zu starten, muss man lediglich *Personal File Sharing* durch Auswahl des entsprechenden Kontrollkästchens oder

[30] Bis zur Version 10.3.2 gab es in Abhängigkeit von der Art des Verbindungsaufbaus subtile Unterschiede beim Verbindungsabbau. Das Update auf mindestens Version 10.3.3 kann in diesem Zusammenhang nur empfohlen werden.

Abb. 3.46. Der AFP-Server kann in den Systemeinstellungen Sharing als *Personal File Sharing* gesteuert werden.

durch einen Klick auf den Start-Knopf aktivieren. Mac OS X startet daraufhin im Hintergrund das Programm `AppleFileServer`, welches alle ankommenden Anfragen von AFP-Clients beantwortet. In der Einzelplatzvariante von Mac OS X und in der Zehn-Platz-Variante von Mac OS X Server ist `AppleFileServer` dabei auf maximal zehn parallele AFP-Verbindungen beschränkt.

Die Einzelplatzvariante enthält keine grafische Benutzerschnittstelle, um die Konfiguration des AFP-Servers zu beeinflussen. Der AFP-Server ist komplett vorkonfiguriert und gibt alle angeschlossenen Datenträger, alle Home-Verzeichnisse und alle öffentlichen Verzeichnisse der lokalen Benutzer frei.

Der Zugriff auf die als Ganzes freigegebenen Datenträger ist dabei ausschließlich lokalen Administratoren, und der Zugriff auf die Home-Verzeichnisse ausschließlich den Eigentümern dieser Verzeichnisse vorbehalten. Auf

diese Weise können Administratoren auch über das Netzwerk auf alle Verzeichnisse zugreifen, auf die sie auch lokal Zugriff hätten, und lokale Benutzer haben die Möglichkeit, über das Netzwerk ihr lokales Home-Verzeichnis zu nutzen.

Zusätzlich kann jeder lokale Benutzer auf die öffentlichen Verzeichnisse (`/Users/kurzname/Public/`) der anderen lokalen Benutzer zugreifen. Nach erfolgreicher Authentifizierung kann ein lokaler Benutzer also nicht nur sein eigenes Home-Verzeichnis aktivieren, sondern auch einen oder mehrere öffentliche Verzeichnisse der anderen lokalen Benutzer des Server-Rechners. Das öffentliche Verzeichnis erscheint dabei in der Liste der verfügbaren Freigaben unter dem Namen des jeweiligen Eigentümers.

Weitere Freigaben einrichten

Apple hat für die Einzelplatzvariante von Mac OS X keine graphische Benutzerschnittstelle für die Einrichtung weiterer Freigaben vorgesehen. Erst die Servervariante von Mac OS X enthält Werkzeuge, mit welchen die Parameter des AFP-Servers beeinflusst und beliebige Freigaben eingerichtet werden können.

Dennoch lassen sich auch in der Einzelplatzvariante von Mac OS X weitere Freigaben einrichten. Da Apple zusätzliche Freigaben nur im Kontext von Mac OS X Server unterstützt, kann man sich natürlich nicht darauf verlassen, dass in der Einzelplatzvariante alle Leistungsmerkmale funktionieren und alle Parameterkombinationen das gewünschte Ergebnis liefern. Diese Art der Konfiguration ist demnach für den Einsatz zu Hause und zum Experimentieren, aber nicht für den Produktiveinsatz in einem Unternehmen geeignet.

Um eigene Freigaben einzurichten, muss man in der lokalen NetInfo-Datenbank per Hand ein Verzeichnis `/config/SharePoints` anlegen. `AppleFileServer` liest dieses Verzeichnis und richtet für jeden Eintrag, den er in diesem Verzeichnis findet, eine AFP-Freigabe ein.

Innerhalb dieses Verzeichnisses muss man demnach für jede Freigabe einen Eintrag erzeugen, dessen Attribute festlegen, welches Verzeichnis wie freigegeben werden soll (siehe Tabelle 3.2).

Um nun z. B. das Verzeichnis `/Users/Shared` (im Finder deutsch `/Benutzer/Für alle Benutzer`) über AFP im Netzwerk freizugeben, muss man im NetInfo-Verzeichnis `/config/SharePoints` einen Eintrag erzeugen, der mindestens die folgenden Attribute enthält:

```
name: Gemeinsame Dateien
directory_path: /Users/Shared
afp_shared: 1
```

Die Bezeichnung der AFP-Freigabe (Abb. 3.47) lässt sich über das Attribut `afp_name` festlegen. Fehlt dieses Attribut, übernimmt Mac OS X 10.4 die im Attribut `name` festgelegte Bezeichnung des NetInfo-Eintrags.

Tabelle 3.2. Attribute von AFP-Freigaben

Attribut	Beschreibung
name	Beliebige Bezeichnung für den NetInfo-Eintrag, z. B. Name des freigegebenen Verzeichnisses.
directory_path	Lokaler Pfad zum freigegebenen Verzeichnis
afp_name	Angepasster AFP-Name. Fehlt diese Angabe, wird der Name des freigegebenen Order verwendet.
afp_shared	1, wenn die Freigabe aktiviert werden soll. Kann auf 0 gesetzt werden, um die Freigabe temporär zu deaktivieren.
afp_guestaccess	1, wenn Gastzugriff möglich sein soll, sonst 0. In der Einzelplatzversion von Mac OS X ohne Auswirkung (Gäste können immer zugreifen).

Unter Mac OS X 10.3 ignorierte der AFP-Server das Attribut `name` und übernahm stattdessen einfach den Namen des freigegebenen Verzeichnisses, im obigen Beispiel also `Shared`. Um unter Mac OS X 10.3 einen vom lokalen Verzeichnisnamen abweichenden Namen für die AFP-Freigabe festzulegen, musste in jedem Fall das Attribut `afp_name` verwendet werden, z. B.:

```
afp_name: Gemeinsame Dateien
```

Mit dem Attribut `afp_use_parent_privs` lässt sich beeinflussen, welche Zugriffsrechte die innerhalb der Freigabe neu angelegten Dateien und Verzeichnisse erhalten sollen.

```
afp_use_parent_privs: 1
```

Fehlt dieses Attribut, bzw. hat es den Wert 0, hängen die Zugriffsrechte der neu angelegten Objekte von den Vorgaben des AFP-Clients ab. Handelt es sich beim AFP-Client um einen Mac-OS-X-Rechner, richten sich die Zugiffsrechte nach der aktiven *umask* (engl. user file-creation mask) des Benutzers. Da die umask in Mac OS X normalerweise den Wert 0022 hat, sind neu angelegte Dateien nur für den Eigentümer lesbar und schreibbar. Alle anderen Benutzer können neu angelegte Dateien lesen, aber nicht verändern:

```
$ umask
0022
$ umask -S
u=rwx,g=rx,o=rx
```

Greifen mehrere Benutzer gemeinsam auf ein Verzeichnis zu, ist dieses Zugriffsrechtemodell oft zu restriktiv. Sollen die von Benutzer A angelegten Objekte von Benutzer B weiterbearbeitet werden können, kann man dieses Problem normalerweise nur umgehen, wenn der Benutzer A entweder jedes Mal manuell die Zugriffsrechte der angelegten Objekte anpasst oder dauerhaft seine umask ändert.

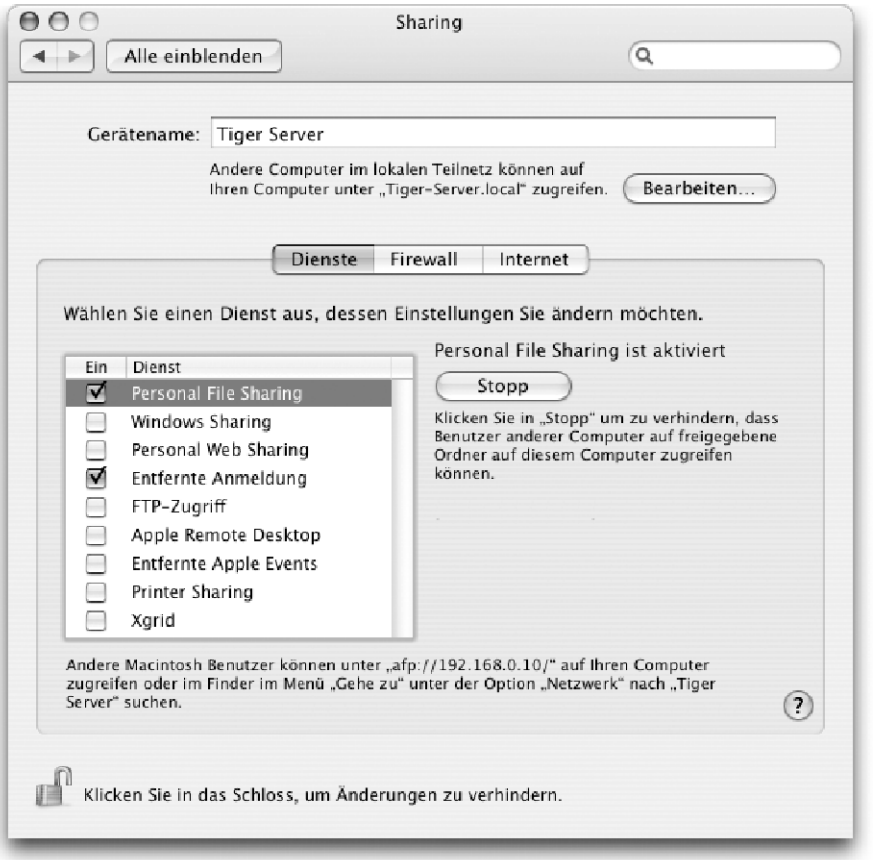


Abb. 3.47. Die Bezeichnung der AFP-Freigabe lässt sich über das Attribut afp_name festlegen. Fehlt dieses Attribut, übernimmt Mac OS X 10.4 die im Attribut name festgelegte Bezeichnung des NetInfo-Eintrags.

Wird das Attribut `afp_user_parent_privs` mit dem Wert 1 belegt, ignoriert der AFP-Server die Vorgaben des AFP-Clients, und übernimmt die Zugriffsrechte einfach vom umschließenden Verzeichnis. Ist das umschließende Verzeichnis für eine bestimmte Gruppe schreibbar, sind demnach auch alle in diesem Verzeichnis neu angelegten Objekte für die Mitglieder dieser Gruppe schreibbar.

Abbildung 3.48 zeigt das vollständige Beispiel im NetInfo-Manager. Nach der Erstellung der gewünschten Einträge in der NetInfo-Datenbank kann der AFP-Server in Systemeinstellungen Netzwerk gestartet werden. Um einen laufenden AFP-Server zum Einlesen einer geänderten Konfiguration zu bewegen, muss man ihn normalerweise anhalten und wieder neu starten. Eine bessere, obgleich nicht dokumentierte Möglichkeit scheint das Senden ei-

Abb. 3.48. Eigene Freigaben können in der NetInfo-Datenbank definiert werden.

nes SIGHUP-Signals zu sein, das ein Einlesen der geänderten Konfiguration im laufenden Betrieb, ohne Unterbrechung von bestehenden Verbindungen, erlaubt.

Da der `AppleFileServer` seine Prozessnummer in der Datei `/var/run/AppleFileServer.pid` ablegt, kann das sowohl mit Hilfe des `kill`-Befehls als auch mit Hilfe des `killall`-Befehls geschehen:

```
$ sudo kill -HUP ‘cat  /var/run/AppleFileServer.pid‘
...
$ sudo killall -HUP AppleFileServer
```

Public-Freigaben deaktivieren

Im Auslieferungszustand gibt der AFP-Server die Public-Verzeichnisse (deutsch „Öffentlich“) aller lokalen Benutzer automatisch frei. Da der AFP-Gastzugang ebenfalls erlaubt ist, kann jeder Benutzer im Netzwerk die im Public-Verzeichnis abgelegten Inhalte lesen. Im Public-Verzeichnis vorkonfi-

guriert ist ein Unterverzeichnis „Drop Box“ (deutsch „Briefkasten“), der für alle Benutzer schreibbar, aber nur für den Eigentümer lesbar ist.

Durch diese Konfiguration ist der direkte Austausch von Dateien sehr einfach zu bewerkstelligen, ohne dass es einer weitergehenden Konfiguration bedürfte. Der lokale Benutzer legt dazu diejenigen Dokumente, die er anderen Benutzern im Netzwerk zur Verfügung stellen möchte, einfach in das Public-Verzeichnis. Andere Benutzer können wiederum Dokumente in der „Drop Box“ ablegen, die für den lokalen Benutzer gedacht sind. Die Erstellung und Verwaltung von Benutzerkennungen und Passwörtern ist nicht erforderlich.

So einfach und vielfach praktisch dieses vordefinierte Schema ist – es gibt Fälle, wo man verhindern möchte, dass das Public-Verzeichnis eines Benutzers automatisch freigegeben wird. Unter Mac OS X 10.3 lässt sich dieser Automatismus dann auch recht einfach abstellen.

Die Information, welches Verzeichnis als Public-Verzeichnis freigegeben werden soll, legt Mac OS X 10.3 zusammen mit allen anderen Attributen der lokalen Benutzer in der NetInfo-Datenbank ab. Für diese automatischen Freigaben verwendet der AFP-Server das Attribut `SharedDir`. Normalerweise enthält dieses Attribut den Wert `Public`, der dem Namen des freizugebenden Verzeichnisses entspricht. Fehlt dieses Attribut, oder enthält es keinen bzw. einen leeren Wert, gibt der AFP-Server das Public-Verzeichnis des entsprechenden Benutzers nicht frei.

Auf diese Weise lässt es sich also im Detail festlegen, welcher Benutzer über ein über AFP freigegebenes Public-Verzeichnis verfügen soll, und welcher nicht.

AppleFileServer Konfigurationsparameter beeinflussen

Die man-Page von `AppleFileServer` gibt keinen Aufschluss darüber, auf welche Weise die Konfiguration des AFP-Servers beeinflussbar ist. Apple hat lediglich die Option `-v` – Ausgabe der Versionsnummer – und die Option `-d` – Betrieb im Vordergrund – dokumentiert.

```
$ AppleFileServer -v
afpserver-500.9
```

Der AFP-Server liest seine Konfiguration aus der Datei `com.apple.AppleFileServer.plist`, die er im Verzeichnis `/Library/Preferences/` erwartet. Fehlt diese Datei, kopiert der AFP-Server beim ersten Start die Datei `PFSDefaults.plist` (Personal File Sharing Defaults?) aus dem Verzeichnis `/System/Library/CoreServices/AppleFileServer.app/Contents/Resources/` nach `/Library/Preferences/`:

```
$ defaults read /Library/Preferences/com.apple.
  AppleFileServer
{
    TCPQuantum = 262144;
    activityLog = 0;
    activityLogPath = "/Library/Logs/AppleFileService/
                        AppleFileServiceAccess.log";
```

```
    activityLogSize = 1000;
    activityLogTime = 7;
    admin31GetsSp = 1;
    adminGetsSp = 0;
    afpTCPPort = 548;
    allowRootLogin = 0;
    attemptAdminAuth = 1;
    authenticationMode = "standard_and_kerberos";
    autoRestart = 1;
    clientSleepOnOff = 1;
    clientSleepTime = 24;
    "enforce_unix_access" = 0;
    errorLogPath = "/Library/Logs/AppleFileService/
                      AppleFileServiceError.log";
    errorLogSize = 1000;
    errorLogTime = 0;
    guestAccess = 1;
    idleDisconnectFlag = {adminUsers = 1; guestUsers = 1;
                          registeredUsers = 1;
                          usersWithOpenFiles = 1; };
    idleDisconnectMsg = "";
    idleDisconnectOnOff = 0;
    idleDisconnectTime = 10;
    kerberosPrincipal = afpserver;
    "lock_manager" = 1;
    loggingAttributes = {
        logCreateDir = 1;
        logCreateFile = 1;
        logDelete = 1;
        logLogin = 1;
        logLogout = 1;
        logOpenFork = 1;
    };
    loginGreeting = "";
    loginGreetingTime = 0;
    noNetworkUsers = 0;
    permissionsModel = "classic_permissions";
    recon1SrvrKeyTTLHrs = 168;
    recon1TokenTTLMins = 10080;
    reconnectFlag = "no_admin_kills";
    reconnectTTLInMin = 1440;
    registerApple\-Talk = 1;
    registerNSL = 1;
    sendGreetingOnce = 0;
    serverStoppedTime = 1122637145;
    shutdownThreshold = 3;
    specialAdminPrivs = 0;
    tickleTime = 30;
    updateHomeDirQuota = 1;
    useApple\-Talk = 0;
}
```

Die innerhalb der Datei `com.apple.AppleFileServer.plist` verwendeten Schlüsselwörter hat Apple im zur Servervariante von Mac OS X gehörenden Handbuch *Command-Line Administration*[31] dokumentiert.

[31] Kapitel „Working with File Services“, S. 82ff, kostenloser Download unter http://www.apple.com/de/server/documentation/ möglich.

Während einige Schlüsselwörter (z. B. `activityLog`) sowohl in der Server- als auch in der Einzelplatzvariante von Mac OS X funktionieren, scheint `AppleFileServer` allerdings andere Schlüsselwörter (z. B. `admin31GetsSp`) in der Einzelplatzversion vollständig zu ignorieren. In Tabelle 3.3 findet man eine Auswahl von Schlüsselwörtern, die auch in der Einzelplatzversion von Mac OS X 10.4 wirksam sind.

Sehr nützlich kann u. U. die Option `acitivityLog` sein, mit der sich das Zugriffsprotokoll aktivieren lässt. Wird das Zugriffsprotokoll aktiviert, erstellt `AppleFileServer` automatisch die in `activityLogPath` angegebene Datei

Tabelle 3.3. Ausgewählte Konfigurationsparameter des AFP-Servers

Parameter	Typ	Beschreibung
`activityLog`	Wahrheitswert	Zugriffsprotokoll aktivieren. In *PFSDefaults.plist* deaktiviert
`activityLogPath`	Zeichenkette	Zugriffsprotokolldatei. In *PFSDefaults.plist* /Library /Logs/ AppleFileService/ AppleFileServiceAccess.log
`errorLogPath`	Zeichenkette	Fehlerprotokolldatei (das Fehlerprotokoll ist immer aktiv). In *PFSDefaults.plist* /Library /Logs/ AppleFileService/ AppleFileServiceError.log.
`logLogin`	Wahrheitswert	FPLogin aufzeichnen. In *PFSDefaults.plist* `true`.
`logLogout`	Wahrheitswert	FPLogout aufzeichnen. In *PFSDefaults.plist* `true`.
`logCreateDir`	Wahrheitswert	FPCreateDir aufzeichnen. In *PFSDefaults.plist* `true`.
`logCreateFile`	Wahrheitswert	FPCreateFile aufzeichnen. In *PFSDefaults.plist* `true`.
`LogDelete`	Wahrheitswert	FPDelete aufzeichnen. In *PFSDefaults.plist* `true`.
`logOpenFork`	Wahrheitswert	FPOpenFork aufzeichnen. In *PFSDefaults.plist* `true`.
`afpTCPPort`	Zahl	TCP Port für AFP über TCP. In *PFSDefaults.plist* 548.
`registerApple\-Talk`	Wahrheitswert	Auffindbarkeit mit AppleTalk/NBP. In *PFSDefaults.plist* `true`
`registerNSL`	Wahrheitswert	Auffindbarkeit mit Bonjour/ZeroConf (Multicast DNS). In *PFSDefaults.plist* `true`
`guestAccess`	Wahrheitswert	Gastzugriff erlauben. In *PFSDefaults.plist* `true`.
`loginGreeting`	Zeichenkette	Begrüßungstext. In *PFSDefaults.plist* leer.
`sendGreetingOnce`	Wahrheitswert	Begrüßungstext nur einmal pro Benutzer senden. In *PFSDefaults.plist* `false`.

und protokolliert dort sowohl die An- und Abmeldung von AFP-Clients als auch deren Zugriff auf freigegebene Dateien:

```
IP 192.168.0.24 - - [23/Aug/2004:12:07:04 0100]
                    "Login rkk" 0 0 0
...
IP 192.168.0.24 - - [23/Aug/2004:12:07:13 0100]
                    "OpenFork projektx.txt" 0 0 0
IP 192.168.0.24 - - [23/Aug/2004:12:07:13 0100]
                    "OpenFork projektx.txt.backup" 0 0 0
...
IP 192.168.0.24 - - [23/Aug/2004:12:07:26 0100]
                    "CreateDir Neuer Ordner" 0 0 0
...
IP 192.168.0.24 - - [23/Aug/2004:12:07:32 0100]
                    "Delete projektx.txt.backup.txt" 0 0 0
...
IP 192.168.0.24 - - [23/Aug/2004:12:07:38 0100]
                    "Logout rkk" 0 0 0
```

Jede Zeile des Zugriffsprotokolls steht dabei für einen einzelnen Zugriff. Protokolliert werden die IP-Adresse des AFP-Clients, der exakte Zeitpunkt des Zugriffs, die Art des Zugriffs bzw. die verwendete AFP-Funktion (FPLogin, FPOpenFork, FPCreateDir, FPCreateFile, FPDelete oder FPLogout), sowie der Rückgabewert der AFP-Funktion. Der Rückgabewert 0 bedeutet, dass der Zugriff erfolgreich abgeschlossen werden konnte, alle anderen Werte deuten auf einen Fehler hin. Eine vollständige Liste der knapp 50 Fehlercodes kann man der aktuellen AFP-Spezifikation von Apple (Titel: *Apple Filing Protocol Version 3.1*)[32] entnehmen. Am häufigsten begegnet man jedoch den in Tabelle 3.4 gelisteten Fehlercodes.

Neben dateibezogenen Zugriffsproblemen lassen sich mit Hilfe des Zugriffsprotokolls also auch nicht erfolgreiche Anmeldeversuche erkennen:

```
...
IP 192.168.0.24 - - [23/Aug/2004:12:13:11 0100]
                    "Login Administrator" -5023 0 0
IP 192.168.0.24 - - [23/Aug/2004:12:13:11 0100]
                    "Logout Administrator" -5023 0 0
...
```

Tabelle 3.4. Typische Fehlercodes des AFP-Servers

Fehlercode	Beschreibung
-5000	Zugriff verweigert. Zugriffsrechte ungenügend.
-5006	Zugriff verweigert. Zugriffssperre aktiv (z. B. Datei bereits durch einen anderen Benutzer geöffnet).
-5007	Verzeichnis ist nicht leer und kann nicht gelöscht werden.
-5023	Login verweigert. Passwort falsch.

[32] http://developer.apple.com/documentation/Networking/Conceptual/AFP/AFP3_1.pdf

Abb. 3.49. Der AFP-Begrüßungstext ist für kurze Mitteilungen an die Benutzer geeignet.

Ebenfalls sehr nützlich ist die Möglichkeit zur Deaktivierung des Gastzugriffs über die Option `guestAccess`. Normalerweise ist in der Einzelplatzversion von Mac OS X der Gastzugriff immer aktiviert. Startet man den AFP-Server, sind damit automatisch sämtliche Public-Verzeichnisse der lokalen Benutzer im Netzwerk sichtbar, ohne dass eine Authentifizierung notwendig wäre. Schaltet man diese Option allerdings auf `false`, ist der Gastzugriff nicht mehr möglich und die Eingabe eines gültigen lokalen Benutzernamens und des dazugehörigen Passworts zwingend erforderlich. Leider lässt sich auf diese Weise der Gastzugang nur global für alle Freigaben des AFP-Servers sperren. Wie bereits festgestellt, erlaubt die Einzelplatzvariante von Mac OS X keine feinere Kontrolle des Gastzugangs, da das Attribut `afp_guestaccess` bei der Definition von Freigaben ignoriert wird. In der Servervariante kann man hingegen den Gastzugang global zulassen und für jede Freigabe einzeln festlegen, ob Gäste Zugriff erhalten sollen oder nicht.

Für kurze Mitteilungen an die Benutzer des AFP-Servers ist schließlich die Option `loginGreeting` geeignet (Abb. 3.49). Benutzern, die eine Verbindung zum AFP-Server erfolgreich aufbauen, wird diese Mitteilung in einer Dialogbox präsentiert. Über die Option `sendGreetingOnce` lässt sich steuern, ob die Mitteilung jedes Mal oder nur ein einziges Mal pro Benutzer angezeigt wird.

SharePoints zur Konfiguration verwenden

Wer davor zurückschreckt, die Voreinstellungsdatei des AFP-Servers und die NetInfo-Datenbank selbst zu editieren, kann auf das Shareware-Programm *SharePoints* von Michael Horn zurückgreifen. *SharePoints* ist eine grafische Benutzeroberfläche zur Manipulation einiger Einstellungen des AFP-Servers (Abb. 3.50) und zur Erstellung von eigenen Freigaben (Abb. 3.51).

Für mit dieser Software eingerichtete Freigaben gelten dann auch genau die gleichen Einschränkungen wie für manuell eingetragene Freigaben. Beide Verfahren stützen sich auf wenig dokumentierte und offiziell nicht unterstüt-

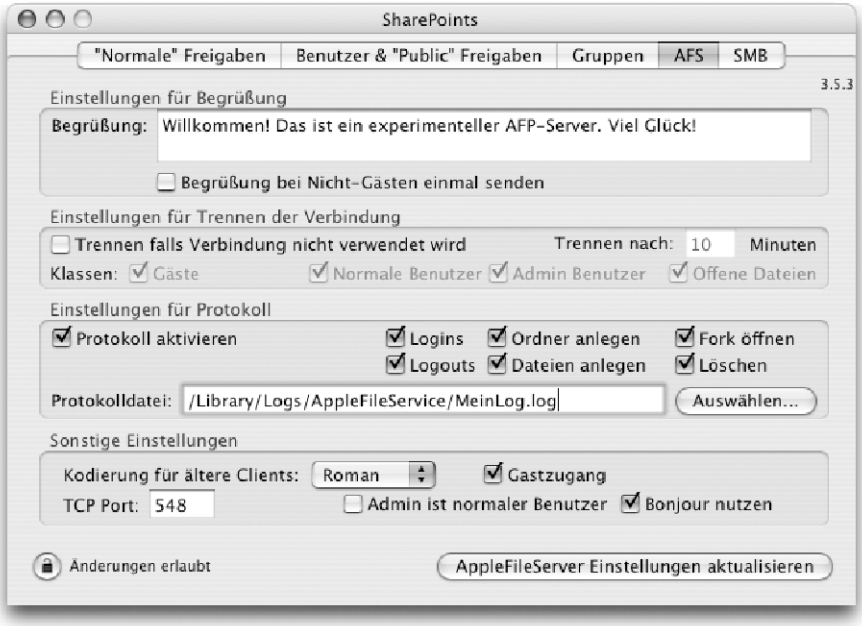


Abb. 3.50. *SharePoints* ist eine grafische Benutzeroberfläche zur Manipulation einiger Einstellungen des AFP-Servers ...

Abb. 3.51. ... und zur Erstellung von eigenen Freigaben in der NetInfo-Datenbank.

zte Leistungsmerkmale der Einzelplatzvariante von Mac OS X und sind daher für den Produktionsbetrieb in einem Unternehmen natürlich nicht geeignet.

3.6.2 SMB-Server aktivieren

Da zum Installationsumfang von Mac OS X auch Samba gehört, kann jeder Mac-OS-X-Rechner problemlos die Rolle eines SMB-Servers übernehmen. Der SMB-Server lässt sich genauso wie der AFP-Server in den Systemeinstellungen Sharing aktivieren (Abb. 3.52).

Anders als bei „Personal File Sharing“ wird unmittelbar nach der Aktivierung von „Windows Sharing“ jedoch kein Hintergrundprozess gestartet. Stattdessen wird unter Mac OS X 10.3 die Konfiguration von `xinetd` und unter Mac OS X 10.4 die Konfiguration von `launchd` dahingehend geändert,

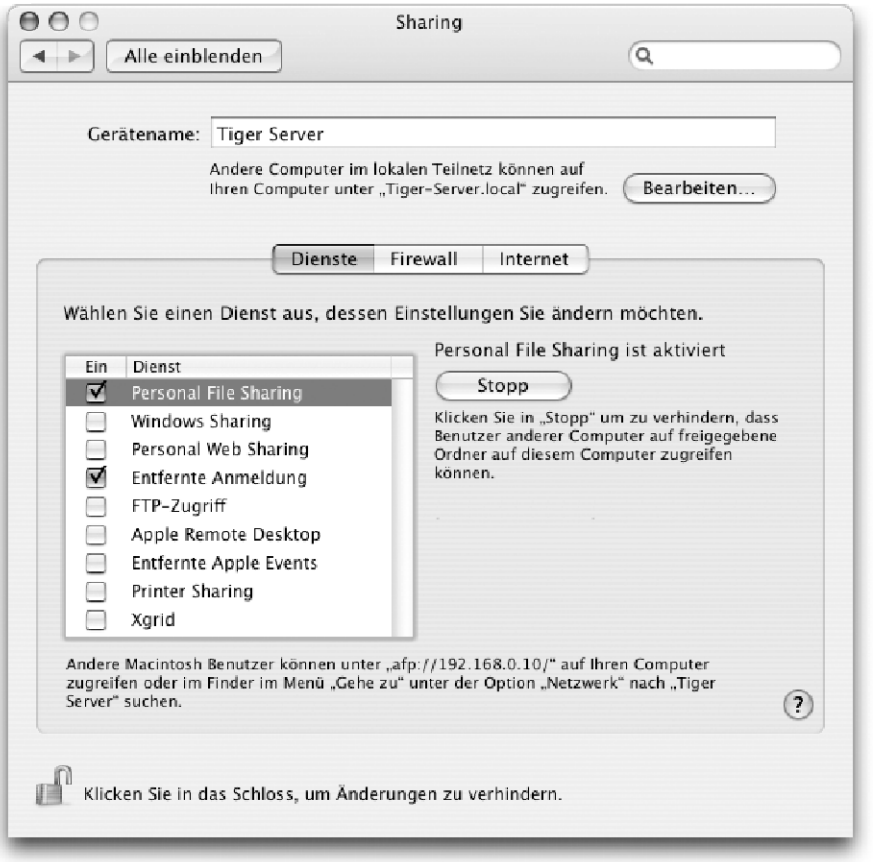


Abb. 3.52. Der SMB-Server lässt sich genauso wie der AFP-Server in den Systemeinstellungen Sharing aktivieren.

dass für die Bearbeitung von Anfragen auf den von SMB verwendeten Ports 137 (NetBIOS-Namensauflösung) und 139 (NetBIOS-Transportprotokoll) jeweils *bei Bedarf* die entsprechenden Hintergrundprozesse gestartet werden.

Da unter Mac OS X 10.3 die entsprechenden Konfigurationsdateien `/etc/xinet.d/nmbd` und `/etc/xinet.d/smbd` bereits existieren, ändert sich durch die Aktivierung von „Windows Sharing“ lediglich der Wert des Attributs `disable` von `yes` auf `no` in diesen beiden Dateien:

```
$ cat /etc/xinetd.d/nmbd
service netbios-ns
{
        disable = no
        socket_type     = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server          = /usr/sbin/nmbd
        groups          = yes
        flags           = REUSE
        log_on_success  =
        log_on_failure  =
}
$ cat /etc/xinetd.d/smbd
service netbios-ssn
{
        disable = no
        socket_type     = stream
        protocol        = tcp
        wait            = no
        user            = root
        server          = /usr/sbin/smbd
        groups          = yes
        flags           = REUSE
        log_on_success  =
        log_on_failure  =
}
```

Unter Mac OS X 10.4 werden durch die Aktivierung von Windows-Sharing die vorkonfigurierten Launch-Daemons geladen:

```
$ sudo launchctl list
...
org.samba.nmbd
org.samba.smbd

$ cat nmbd.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>org.samba.nmbd</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/nmbd</string>
        </array>
```

```
        <key>Sockets</key>
        <dict>
                <key>Listeners</key>
                <dict>
                        <key>SockFamily</key>
                        <string>IPv4</string>
                        <key>SockServiceName</key>
                        <string>netbios-ns</string>
                        <key>SockType</key>
                        <string>dgram</string>
                </dict>
        </dict>
        <key>inetdCompatibility</key>
        <dict>
                <key>Wait</key>
                <true/>
        </dict>
</dict>
</plist>

$ cat smbd.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>org.samba.smbd</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/smbd</string>
        </array>
        <key>Sockets</key>
        <dict>
                <key>direct</key>
                <dict>
                        <key>SockFamily</key>
                        <string>IPv4</string>
                        <key>SockServiceName</key>
                        <string>microsoft-ds</string>
                </dict>
                <key>netbios</key>
                <dict>
                        <key>SockFamily</key>
                        <string>IPv4</string>
                        <key>SockServiceName</key>
                        <string>netbios-ssn</string>
                </dict>
        </dict>
        <key>inetdCompatibility</key>
        <dict>
                <key>Wait</key>
                <false/>
        </dict>
</dict>
</plist>
```

Ähnlich wie beim AFP-Server hat Apple auch für den SMB-Server zunächst keine weiteren Konfigurationsmöglichkeiten vorgesehen. Im Aus-

lieferungszustand ermöglicht der SMB-Server jedem lokalen Benutzer den Zugriff auf sein Home-Verzeichnis, der Zugriff auf die Public-Verzeichnisse der anderen Benutzer ist aber genauso wenig möglich wie der Zugriff auf die angeschlossenen Datenträger als Ganzes.

Da es sich jedoch beim SMB-Server um einen Standard-Samba-Server handelt, lässt er sich aber auch entsprechend flexibel konfigurieren und einsetzen. Gleichzeitig kann man bei der Konfiguration des SMB-Servers auf eine umfangreiche mitgelieferte Dokumentation zurückgreifen. Nicht nur, dass die meisten Komponenten von Samba über eine aussagekräftige man-Page verfügen – im Verzeichnis `/usr/share/swat/using_samba/` findet man die vollständige englischsprachige Ausgabe des Buchs „Using Samba“ (O'Reilly) im HTML-Format (Abb. 3.53).

Die Samba-Voreinstellungsdatei

Die beiden Hintergrundprozesse, `nmbd` und `smbd`, die den SMB-Server realisieren, werden mit Hilfe der Voreinstellungsdatei `/etc/smb.conf` gesteuert.

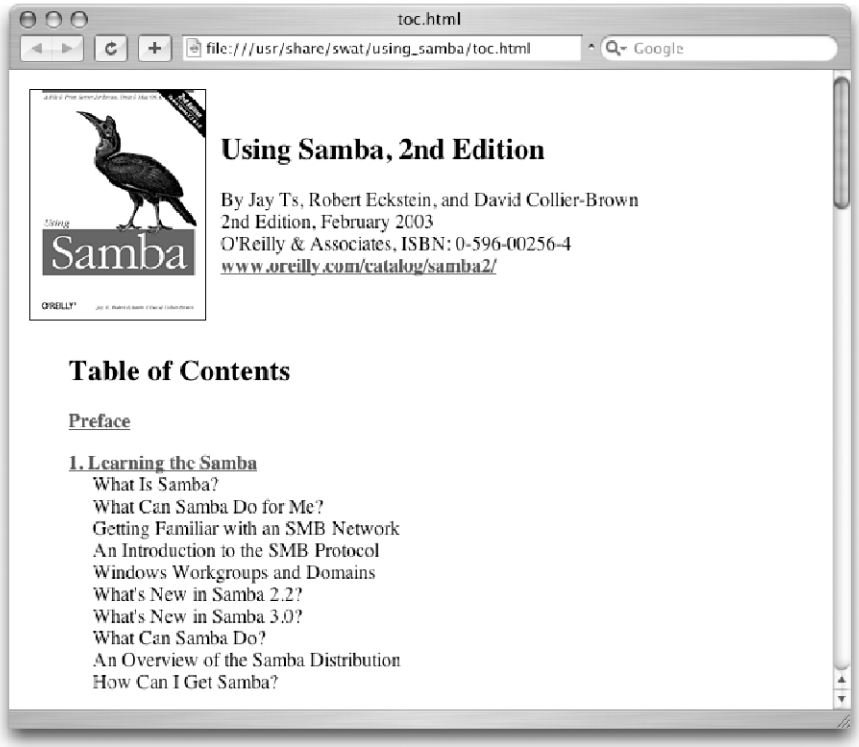


Abb. 3.53. Auf der lokalen Festplatte findet man die vollständige englischsprachige Ausgabe des Buchs „Using Samba“ (O'Reilly) im HTML-Format.

Das Format von `smb.conf` ähnelt den auf Windows-Systemen gebräuchlichen *.ini*-Dateien. In Mac OS 10.4 hat diese Datei normalerweise folgendes Aussehen:

```
$ cat /etc/smb.conf
; Template configuration file for smbd.
; ======================================================
; For the format of this file and comprehensive descriptions
; of all the configuration option, please refer to the man
; page for smb.conf(5).
;
; The following configuration should suit most systems for
; basic usage and initial testing. It gives all clients
; access to their home directories and allows access to all
; printers specified in /etc/printcap.  It also provides
; a public share point for generally exporting stuff.
;
; Some things to check out:
;
; 1: Make sure that the user specified in "guest account"
;    exists. Typically this will be a user that cannot log
;    in and has minimal privileges. Often the "nobody"
;    account doesn't work (very system dependant).
;
; 2: You should consider the "security =" option. See a full
;    description in the main documentation and the
;    smb.conf(5) manual page
;
; 3: Look at the "hosts allow" option, unless you want
;    everyone on the internet to be able to access the
;    shares you export here.
;
; 4: If you want to support printers, add/uncomment the
;    relevant entries.
;
[global]
  guest account = unknown
  encrypt passwords = yes
  auth methods = guest opendirectory
  passdb backend = opendirectorysam guest
  printer admin = @admin, @staff
  server string = Tiger Server
  unix charset = UTF-8-MAC
  display charset = UTF-8-MAC
  dos charset = 437
  use spnego = yes
  client ntlmv2 auth = no
  os level = 8
  defer sharing violations = no
  vfs objects = darwin_acls
  brlm = yes
  workgroup = WORKGROUP
; Using the Computer Name to compute the NetBIOS name.
; Remove this comment to override
  netbios name = Tiger-Server
[homes]
   comment = User Home Directories
   browseable = no
   read only = no
```

```
;[public]
;   path = /tmp
;   public = yes
;   only guest = yes
;   writable = yes
;   printable = no

[printers]
  path = /tmp
  printable = yes
```

Die Voreinstellungsdatei wird durch die in eckigen Klammern angegebenen Ausdrücke in Abschnitte unterteilt. Alle Zeilen, die mit einem `;` (oder einem `#`) beginnen, werden als Kommentarzeilen interpretiert und vom SMB-Server entsprechend ignoriert.

Im Abschnitt `[global]` stehen Voreinstellungen des Servers selbst, oder Voreinstellungen, die für alle Freigaben gelten sollen. Der Abschnitt `[homes]` definiert für jeden Benutzer des Systems sein jeweiliges Home-Verzeichnis als eine über SMB zugängliche Freigabe. Der Abschnitt `[printers]` definiert für jeden an das System angeschlossenen Drucker eine Druckerfreigabe. Der (auskommentierte) Abschnitt `[public]` ist als Beispiel gedacht – würde man die Kommentarzeichen entfernen, würde dieser Abschnitt den SMB-Server anweisen, das Verzeichnis `/tmp` als ein für alle zugängliches (`public = yes`) und schreibbares (`writable = yes`) Verzeichnis freizugeben.

Samba liest die Voreinstellungsdatei alle 60 Sekunden – man kann den beiden Samba-Daemons aber auch einfach ein HUP-Signal schicken, um das sofortige Einlesen der Datei zu veranlassen:

```
$ sudo killall -HUP smbd nmbd
```

NetBIOS konfigurieren

Um als vollwertiger SMB-Server arbeiten zu können, benötigt ein Mac-OS-X-Rechner einen NetBIOS-Namen und den Namen seiner Arbeitsgruppe. Soll für die Registrierung des eigenen Namens und für die Abbildung von fremden Namen auf IP-Adressen ein NBNS- bzw. WINS-Server verwendet werden, so muss auch dessen IP-Adresse angegeben werden. Diese Angabe ist jedoch optional.

In Samba wird der NetBIOS-Name normalerweise unmittelbar aus dem DNS-Namen abgeleitet. Ist der Name zu lang, wird er auf die für NetBIOS-Namen maximal zulässige Länge von 15 Zeichen gekürzt. Letzteres passiert in Mac OS X besonders dann häufig, wenn private IP-Adressen ohne einen DNS-Server verwendet werden: in diesem Fall leitet Mac OS X einen DNS-Namen für die Verwendung mit Bonjour/ZeroConf automatisch aus dem AppleTalk/NBP-Gerätenamen (festzulegen in Systemeinstellungen Sharing) ab.

Da der Gerätename als NBP-Name bis zu 32 Zeichen lang sein darf, ist ein daraus durch einfaches Abschneiden abgeleiteter NetBIOS-Name nicht immer ideal. Ein eigenständiger NetBIOS-Name kann in solchen Fällen in

`smb.conf` jedoch jederzeit mit Hilfe der globalen Option `netbios name` angegeben werden, z. B.

```
[global]
...
netbios name = MRHYDE
...
```

Der Name der Arbeitsgruppe und die IP-Adresse des WINS-Servers werden ebenfalls im globalen Abschnitt von `smb.conf` mit Hilfe der Optionen `workgroup` und `wins server` festgelegt:

```
[global]
...
workgroup = KOBYLINSKI
wins server = 192.168.0.2
...
```

Die Werte dieser beiden Optionen gibt man jedoch am besten über das Dienstprogramm Verzeichnisdienste (Abb. 3.54) ein. Die dort eingegebenen Werte werden automatisch in `smb.conf` eingetragen. Umgekehrt liest das Dienstprogramm Verzeichnisdienste die in `smb.conf` vorgenommenen Änderungen dieser Werte.

Abb. 3.54. Die Arbeitsgruppe und die IP-Adresse des WINS-Servers kann man im Dienstprogramm Verzeichnisdienste angeben.

Steht ein WINS-Server nicht zur Verfügung, kann der Mac-OS-X-Rechner dank Samba diese Rolle auch selbst übernehmen. Um den Samba-WINS-Server zu aktivieren, genügt es, die globale Option `wins support` in `smb.conf` einzufügen:

```
[global]
...
wins support = yes
...
```

Der Samba-WINS-Server ersetzt dann einen entsprechend konfigurierten Windows NT/2000-Rechner. Die einzige Einschränkung: der Samba-WINS-Server kann seine Daten nicht automatisch mit anderen WINS-Servern abgleichen, so dass man auf den Betrieb von weiteren WINS-Servern verzichten sollte.

Weitere Freigaben einrichten

Samba-Freigaben definiert man, indem man in `smb.conf` für jede Freigabe einen eigenen Abschnitt einfügt und dort angibt, welches Verzeichnis freigegeben werden soll.

Um das Verzeichnis `/Users/Shared` als „Gemeinsames“ freizugeben, genügt es also, in `smb.conf` die folgenden Zeilen einzufügen:

```
[Gemeinsames]
path = /Users/Shared
```

Bei der Benennung der SMB-Freigaben sollte man sich aus Gründen der Kompatibilität zu einer möglichst breiten Palette von SMB-Clients auf maximal 13 Zeichen beschränken. So ist es insbesondere auch unter Mac OS X 10.3 nicht möglich, mit dem Finder eine Verbindung zu einer SMB-Freigabe mit dem (langen) Namen „Gemeinsame Dateien“ herzustellen. Mit dem Mac-OS-X-10.4-Finder hingegen schon.

Ein auf diese Weise freigegebenes Verzeichnis kann nur gelesen, aber nicht verändert werden. Um Schreibzugriffe zu erlauben, muss man die Definition der Freigabe um die Option `read only = no` bzw. die Option `writable = yes`[33] ergänzen:

```
[Gemeinsames]
path = /Users/Shared
writable = yes
```

Protokollierung konfigurieren

Die Aktivitäten der beiden Samba-Prozesse `nmbd` und `smbd` werden automatisch protokolliert. Als Protokolldateien verwendet `nmbd` dabei die Datei `/var/log/samba/log.nmbd` und `smbd` die Datei `/var/log/samba/log.nmbd`.

[33] Eine der beiden Optionen genügt.

Bereits im Auslieferungszustand protokolliert `smbd` die wichtigsten Ereignisse, so dass man dem Protokoll entnehmen kann, wer wann welche Freigaben benutzt hat:

```
...
[2004/08/24 19:04:15, 0] /SourceCache/samba/samba-56/samba/source/
                         smbd/server.c:main(747)
  smbd version 3.0.2 started.
  Copyright Andrew Tridgell and the Samba Team 1992-2004
[2004/08/24 19:04:17, 0] /SourceCache/samba/samba-56/samba/source/
                         smbd/server.c:main(747)
  smbd version 3.0.2 started.
  Copyright Andrew Tridgell and the Samba Team 1992-2004
[2004/08/24 19:04:17, 1] pdb_ods.c:odssam_getsampwrid(1831)
  odssam_getsampwrid: rid<501> rid str<501>
[2004/08/24 19:04:33, 0] /SourceCache/samba/samba-56/samba/source/
                         smbd/server.c:main(747)
  smbd version 3.0.2 started.
  Copyright Andrew Tridgell and the Samba Team 1992-2004
[2004/08/24 19:04:33, 0] /SourceCache/samba/samba-56/samba/source/
                         smbd/server.c:main(747)
  smbd version 3.0.2 started.
  Copyright Andrew Tridgell and the Samba Team 1992-2004
[2004/08/24 19:04:34, 1] auth_ods.c:opendirectory_auth_user(206)
  User "rkk" authenticated successfully with
  "dsAuthMethodStandard:dsAuthSMBNTKey" :)
[2004/08/24 19:04:34, 0] auth_ods.c:
                         opendirectory_smb_pwd_check_ntlmv1(266)
  opendirectory_smb_pwd_check_ntlmv1: [0]opendirectory_auth_user
[2004/08/24 19:04:34, 1] /SourceCache/samba/samba-56/samba/source/
                         smbd/service.c:make_connection_snum(705)
  imac (192.168.0.24) connect to service Gemeinsames initially as
                      user rkk (uid=502, gid=502) (pid 1246)
[2004/08/24 19:04:39, 1] /SourceCache/samba/samba-56/samba/source/
                         smbd/service.c:close_cnum(887)
  imac (192.168.0.24) closed connection to service Gemeinsames
...
```

Sollten diese Informationen einmal nicht ausreichen, kann die Anzahl der protokollierten Detailinformationen mit Hilfe der globalen Option `log level` erhöht werden. Stufe 0 deaktiviert die Protokollierung, Stufe 1 entspricht dem Auslieferungszustand, und höhere Stufen (bis maximal 10) aktivieren entsprechend detailliertere Statusmeldungen. Die Stufe 3 ist allerdings schon recht ausführlich und sollte für die meisten Fälle genügen.

Um ein stetiges Anwachsen der Protokolldatei gerade nach Aktivierung der höheren Protokollierungsstufen zu verhindern, kann man der Option `max log size` eine obere Grenze für die Protokolldateigröße (in Kilobyte) angeben. Ist diese Grenze erreicht, hängt Samba die Endung *.old* an den Namen der bisherigen Protokolldatei an[34] und fängt automatisch eine neue an.

Dateizugriffe werden unabhängig von der Protokollierungsstufe normalerweise nicht mitprotokolliert. Für sie sieht Samba (ab der Version 3.0) zwei

[34] Die alte Protokolldatei wird anschließend also nur so lange aufgehoben, bis die Grenze erneut erreicht wird. Dann wird sie mit der neueren Protokolldatei überschrieben.

Virtual-File-System (kurz VFS)-Module vor: `audit` und `extd_audit`. VFS-Module stellen einen flexiblen Erweiterungsmechanismus dar. Sobald eines oder mehrere VFS-Module installiert sind, leitet Samba alle Dateizugriffe auf das erste installierte VFS-Modul um. Sind mehrere Module installiert, werden Dateizugriffe durch alle Module durchgeleitet, bis schließlich das letzte Modul direkt das Dateisystem berührt.

Die beiden Protokollierungsmodule erstellen für bestimmte Dateizugriffsoperationen einen Protokolleintrag und leiten die Dateizugriffe ansonsten unverändert weiter. Sie unterscheiden sich vornehmlich dahingehend, dass `audit` für die Protokollierung ausschließlich `syslogd` benutzt (wodurch die Einträge in `/var/log/system.log` landen, und `extd_audit` zusätzlich auch noch die normale `smbd`-Protokolldatei verwendet.

Um ein VFS-Modul für eine Freigabe zu installieren, muss man im Freigabebereich die Option `vfs options` einfügen. Um Zugriffe auf die Freigabe „Gemeinsames“ möglichst ausführlich zu protokollieren, müsste man sie wie folgt definieren:

```
[Gemeinsames]
path = /Users/Shared
writable = yes
log level = 3
vfs options = extd_audit
```

SWAT aktivieren

Das Samba-Paket enthält eine vollständige Web-Konfigurationsanwendung namens SWAT (engl. Samba Web Administration Tool). Unter Mac OS X ist SWAT zwar bereits installiert, der Benutzer muss allerdings einige Konfigurationsschritte manuell durchführen, bevor sie tatsächlich zur Verfügung steht.

Im ersten Schritt sollte man überprüfen, ob die Datei `/etc/services` bereits einen Eintrag für SWAT enthält, oder ob ein solcher Eintrag erst angelegt werden muss. SWAT verwendet für die Kommunikation mit dem Web-Browser den TCP-Port 901. In `/etc/services` ist in Mac OS X 10.4 bereits ein Eintrag für diesen Port vorhanden, weist diesem Port jedoch einen anderen Dienst zu.

Um SWAT zu nutzen, muss man also einen zusätzlichen Eintrag für SWAT einfügen. Den Eintrag für das `smpnameres`-Protokoll kann man auskommentieren. Nach dieser Änderung hat `/etc/services` an der geänderten Stelle das folgende Aussehen:

```
...
#smpnameres      901/udp     # SMPNAMERES
#smpnameres      901/tcp     # SMPNAMERES
#                          Leif Ekblad <leif@rdos.net>
swat            901/tcp     # Samba SWAT
...
```

Im zweiten Schritt startet man SWAT. Unter Mac OS X 10.3 verwendete man dafür `xinetd`, unter Mac OS X 10.4 verwendet man stattdessen `launchd`.

Unter Mac OS X 10.3 muss man `xinetd` anweisen, SWAT bei Bedarf zu starten. Eine entsprechende Konfigurationsdatei existiert bereits:

```
$ cat /etc/xinetd.d/swat
service swat
{
        port    = 901
        socket_type     = stream
        wait    = no
        only_from = localhost
        user    = root
        server  = /usr/sbin/swat
        log_on_failure  += USERID
        groups = yes
        disable = yes
}
```

Das Attribut `only_from = localhost` stellt dabei sicher, dass ausschließlich lokale Verbindungen zugelassen werden. Ein Benutzer auf einem anderen Rechner hat damit also keine Möglichkeit, die Samba-Konfiguration mit Hilfe von SWAT zu ändern.

Damit `xinetd` SWAT bei Bedarf auch startet, muss man das Attribut `disable = yes` in `disable = no` ändern. Damit `xinetd` anschließend die geänderte Konfiguration auch berücksichtigt, muss man ihn mit einem HUP-Signal neu initialisieren:

```
$ killall -HUP xinetd
```

Unter Mac OS X 10.4 muss man SWAT als Launch Daemon laden. Die Konfigurationsdatei existiert bereits:

```
$ cat swat.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
             "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Disabled</key>
        <true/>
        <key>GID</key>
        <integer>0</integer>
        <key>Label</key>
        <string>org.samba.swat</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/swat</string>
                <string>-d 10</string>
        </array>
        <key>Sockets</key>
        <dict>
                <key>Listeners</key>
                <dict>
                        <key>SockNodeName</key>
                        <string>localhost</string>
                        <key>SockServiceName</key>
```

```
                        <integer>901</integer>
                </dict>
        </dict>
        <key>inetdCompatibility</key>
        <dict>
                <key>Wait</key>
                <false/>
        </dict>
</dict>
</plist>
```

Anders als unter Mac OS X 10.3 muss sie jedoch nicht per Hand modifiziert werden, sondern kann direkt geladen werden. Der Parameter `-w` entfernt dabei automatisch das Attribut `Disabled` aus der Konfigurationsdatei.

```
$ sudo launchctl load -w /System/Library/LaunchDaemons/
                         swat.plist
```

Wurde die Konfigurationsdatei erfolgreich geladen, erscheint SWAT in der Liste der aktiven `launchd`-Dienste:

```
$ sudo launchctl list
...
org.samba.nmbd
org.samba.smbd
org.samba.swat
```

Um SWAT zu benutzen, öffnet man einfach die lokale URL `http://localhost:901` in einem Web-Browser. Je nachdem, ob man die Samba-Konfiguration verändern oder nur betrachten will, muss man sich mit der lokalen root-Kennung oder mit einer beliebigen anderen lokalen Benutzerkennung authentifizieren.

Nach erfolgreicher Authentifizierung gelangt man zur SWAT-Einstiegsseite (Abb. 3.55), von der man einerseits die verschiedenen Konfigurationsbereiche und andererseits die umfangreiche Dokumentation[35] ansteuern kann. Zur letzteren zählen neben allen relevanten man-Pages auch zwei umfangreiche Bücher – *Using Samba* (Abb. 3.53) und *The Samba HOWTO Collection*.

In den neueren Versionen von Mac OS X scheint die Authentifizierung nicht mehr weitergepflegt worden zu sein. Sollte die Authentifizierung daher nicht klappen, kann man die Konfigurationsdateien von `xinetd` bzw. von `launchd` dahingehend ändern, dass `swat` immer zusammen mit der Option `-a` aufgerufen wird. Die Option `-a` bewirkt die Abschaltung der Authentifizierung, so dass jeder lokale Benutzer die SWAT-Webanwendung aufrufen und die Samba-Konfiguration verändern kann.

3.6.3 NFS-Server aktivieren

Mac OS X ist für den Einsatz als NFS-Server bereits vorkonfiguriert. Die dafür notwendigen Prozesse werden allerdings beim Systemstart nur dann ge-

[35] Die Dokumentation ist im Verzeichnis /usr/share/swat abgelegt und lässt sich damit natürlich auch jederzeit ohne SWAT betrachten.

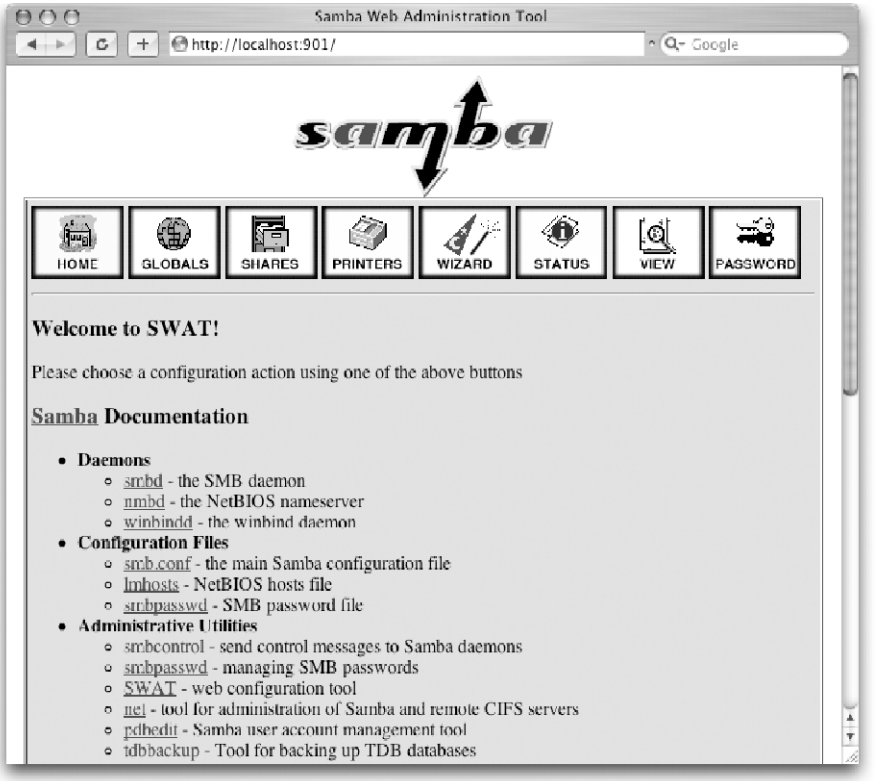


Abb. 3.55. Die Einstiegsseite von SWAT.

startet, wenn Freigaben bereits definiert worden sind. Werden die Freigabendefinitionen erst im laufenden Betrieb erstellt, muss man entweder einen Systemneustart durchführen oder die entsprechenden Prozesse per Hand starten.

Die NFS-Server-Funktionalität wird unter Mac OS X mit Hilfe von drei Prozessen realisiert: `portmap`, `mountd` und `nfsd`.

`portmap` ist ein zentraler Bestandteil des für die Implementierung von NFS verwendeten RPC (engl. Remote Procedure Call)-Protokolls. Jedes RPC-Programm stellt unter einer Programmnummer eine Anzahl von Prozeduren zur Verfügung. Die Datei `/etc/rpc` enthält eine Liste von RPC-Programmen und deren Programmnummern:

```
$ cat /etc/rpc
#
# $FreeBSD: src/etc/rpc,v 1.7 1999/08/27 23:23:44 peter Exp $
# rpc 88/08/01 4.0 RPCSRC; from 1.12   88/02/07 SMI
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
```

```
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
rquotad         100011  rquotaprog quota rquota
sprayd          100012  spray
3270_mapper     100013
rje_mapper      100014
selection_svc   100015  selnsvc
database_svc    100016
rexd            100017  rex
alis            100018
sched           100019
llockmgr        100020
nlockmgr        100021
x25.inr         100022
statmon         100023
status          100024
bootparamd      100026  bootparam
ypupdated       100028  ypupdate
keyserv         100029  keyserver
tfsd            100037
nsed            100038
nsemntd         100039
pcnfsd          150001  pcnfs
amd             300019
cmsd            100068
ttdbserver      100083  tooltalk
#
# The range 200100000-200199999 is reserved for Apple services
#
netinfo       200100000
netinfobind   200100001
```

Da RPC-Programme für die Kommunikation TCP und UDP verwenden, müssen diese Programmnummern zur Laufzeit auf die verwendeten TCP- und UDP-Portnummern abgebildet werden. Dazu muss jeder RPC-Server beim Start seine Programmnummer zusammen mit der aktuell verwendeten Portnummer beim lokalen `portmap`-Prozess registrieren. Ein RPC-Client hat auf diese Weise eine fest definierte Anlaufstelle, bei der er nachfragen kann, welche Programme unter welcher Portnummer aktuell ansprechbar sind.

Mac OS X 10.4 startet `portmap` bei Bedarf über `launchd`:

```
 cat /System/Library/LaunchDaemons/com.apple.portmap.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
             "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>com.apple.portmap</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/sbin/portmap</string>
```

```
        </array>
        <key>Sockets</key>
        <dict>
                <key>OnDemandTickler</key>
                <dict>
                        <key>SockPathName</key>
                        <string>/var/run/portmap.socket</string>
                </dict>
        </dict>
        <key>ServiceIPC</key>
        <true/>
</dict>
</plist>
```

Mit Hilfe des Befehls `rpcinfo` lässt sich sehr einfach überprüfen, ob `portmap` bereits läuft oder erst gestartet werden muss. Wird `rpcinfo` mit der Option `-p` aufgerufen, erhält man, falls `portmap` noch nicht läuft, eine entsprechende Fehlermeldung. Ansonsten wird die Liste aller registrierten RPC-Programm- und Portnummern ausgegeben:

```
$ rpcinfo -p
rpcinfo: can't contact portmapper: RPC: Remote system error -
                                        Connection refused
$ sudo $ sudo launchctl start com.apple.portmap
$ rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
```

Als Nächstes müssen die beiden eigentlichen NFS-Serverprozesse `mountd` und `nfsd` gestartet werden. Mac OS X startet auch diese beiden Prozesse beim Systemstart automatisch, sofern NFS-Freigaben existieren. Die entsprechenden Anweisungen dazu findet man im StartupItem NFS, u. a. z. B.:

```
$ cat /System/Library/StartupItems/NFS/NFS
...
    # If exportfs finds something to export (either using
    # /etc/exports or the exports NetInfo directory), then
    # start the NFS daemons (which service NFS requests) and
    # the mount server (which services NFS mount requests).

    # Clear the table of exported filesystems.
    rm -f /var/db/mountdtab

    if [ "${exports}" -gt 0 ]; then
        echo "Starting Network File System server"
        mountd

        # If the NetInfo config/nfsd directory contains
        # startup args for nfsd, use those.
        arguments=`niutil -readprop . /config/nfsd
                    arguments`
        if [ "${arguments}" = "" ]; then
                arguments="-t -u -n 6"
        fi
        nfsd ${arguments}
    fi
...
```

Sobald nur eine Freigabe eingerichtet wurde, werden also beim Neustart alle notwendigen Daemons automatisch gestartet. Dazu gehören neben `portmap`, `mountd` und `nfsd` auch `rpc.statd` sowie `rpc.lockd`:

```
$ ps ax -o pid -o command | egrep 'portmap|rpc|mountd|nfsd'
  133 rpc.statd
  135 rpc.lockd
  136 /usr/sbin/portmap
  140 rpc.lockd
  154 mountd
  157 nfsd-master
  160 nfsd-server
  161 nfsd-server
  162 nfsd-server
  163 nfsd-server
  164 nfsd-server
  165 nfsd-server
  282 egrep portmap|rpc|mountd|nfsd
```

Mit `rpcinfo` lässt sich nach einer erfolgreichen Aktivierung von NFS verifizieren, dass alle diese Prozesse bei `portmap` registriert wurden:

```
$ rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp   1019  status
    100024    1   tcp   1015  status
    100021    0   udp   1008  nlockmgr
    100021    1   udp   1008  nlockmgr
    100021    3   udp   1008  nlockmgr
    100021    4   udp   1008  nlockmgr
    100021    0   tcp   1014  nlockmgr
    100021    1   tcp   1014  nlockmgr
    100021    3   tcp   1014  nlockmgr
    100021    4   tcp   1014  nlockmgr
    100005    1   udp    989  mountd
    100005    3   udp    989  mountd
    100005    1   tcp   1012  mountd
    100005    3   tcp   1012  mountd
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
```

Freigaben einrichten

Um eine NFS-Freigabe einzurichten, müssen wir in der lokalen NetInfo-Datenbank ein Verzeichnis `/exports` anlegen und dort die einzelnen Freigaben definieren. Eine Freigabendefinition entspricht einem Eintrag innerhalb von `/exports` und definiert bereits mit dem obligatorischen Attribut `name` den Pfad zum freizugebenden Verzeichnis.

Um das Verzeichnis `/Users/Shared` mit Hilfe von NFS freizugeben, genügt es also, in der NetInfo-Datenbank unter `/exports` einen Eintrag mit dem Namen `/Users/Shared` anzulegen (Abb. 3.56).

Abb. 3.56. Ein einfacher Eintrag in /exports genügt, um ein Verzeichnis mit Hilfe von NFS freizugeben.

Anschließend muss nur noch `mountd` durch ein HUP-Signal auf die Änderung der Freigabendefinitionen aufmerksam gemacht werden:

```
$ sudo killall -HUP mountd
```

Handelt es sich um die erste NFS-Freigabe, ist ein Neustart der einfachste Weg, alle notwendigen Prozesse starten zu lassen. In jedem Fall lässt sich anschließend mit dem `showmount`-Befehl überprüfen, ob die Freigabe verfügbar ist:

```
$ showmount -e
Exports list on localhost:
/Users/Shared                       Everyone
```

Eine solche Freigabe ist grundsätzlich für alle Welt lesbar und schreibbar. Da die Authentifizierung im Vertrauen auf eine übereinstimmende Konfiguration der Benutzerdatenbank des NFS-Servers und der NFS-Clients aus-

schließlich anhand der Benutzernummern erfolgt, ist das jedoch oft nicht wünschenswert.

Mit Hilfe von weiteren Optionen lässt sich der Zugriff auf bestimmte Netzwerke oder bestimmte Clients beschränken. Ebenso lässt sich steuern, ob überhaupt Schreibzugriff gewährt werden soll, und in einem gewissen Rahmen lässt sich auch die Abbildung von Benutzernummern beeinflussen.

Um den Zugriff auf bestimmte Hosts einzuschränken, muss man die Freigabedefinition in NetInfo `/exports` um das Attribut `clients` ergänzen. Als Werte dieses Attributs können eine oder mehrere IP-Adressen (oder Namen) von Hosts[36] angegeben werden, denen der Zugriff auf die NFS-Freigabe gestattet werden soll. Allen anderen Hosts wird auf diese Weise implizit der Zugriff verweigert. Im folgenden Beispiel wird der Zugriff auf die beiden Hosts mit den IP-Adressen 192.168.0.2 und 192.168.0.24 eingeschränkt.

```
$ nicl . -read /exports/\\/Users\\/Shared
clients: 192.168.0.2 192.168.0.24
name: /Users/Shared
```

Um einem kompletten Subnetz Zugriff zu gewähren, muss man aber nicht alle IP-Adressen auflisten. Stattdessen kann man die beiden Optionen `network` und `mask` angeben und so den Zugriff auf ein bestimmtes Netzwerk beschränken. Optionen werden über das Attribut `opts` definiert, wobei als Werte alle in der export-man-Page dokumentierten Optionen zugelassen sind.

Um den Zugriff auf das Netzwerk 192.168.0.0/24 einzuschränken, sind demnach die Werte `network=192.168.0` und `mask=255.255.255.0` anzugeben:

```
$ nicl . -read /exports/\\/Users\\/Shared
opts: network=192.168.0 mask=255.255.255.0
name: /Users/Shared
```

Gleichzeitig lassen sich natürlich auch noch weitere Optionen angeben. Nützlich ist z. B. die Option `ro` mit der der Schreibzugriff verhindert werden kann. Ebenso benutzt man häufig die Option `maproot=nobody`, um die Zugriffsrechte eines auf einem NFS-Client als root authentifizierten Benutzers serverseitig einzuschränken:

```
$ nicl . -read /exports/\\/Users\\/Shared
clients: 192.168.0.2 192.168.0.24
name: /Users/Shared
opts: ro maproot=nobody
```

Weitere Freigaben lassen sich grundsätzlich mit Hilfe von zusätzlichen Einträgen in `/exports` definieren. Eine Ausnahme bilden hierbei jedoch mehrere Freigaben, die auf dem gleichen Dateisystem definiert sind und ein leeres

[36] Anwender des Verzeichnisdienstes NIS können hier auch Gruppen von Hosts als Netgroups angeben.

`clients`-Attribut haben, d. h. für alle Welt schreibbar sind. Alle solche Freigaben müssen mit Hilfe eines einzigen Eintrags in `/exports` definiert werden, indem dem Attribut `name` mehrere Werte zugewiesen werden:

```
$ nicl . -read /exports/\\/Users\\/Shared
clients:
name: /Users/rkk /Users/Shared
opts: ro maproot=nobody
$ showmount -e
Exports list on localhost:
/Users/rkk                          Everyone
/Users/Shared                       Everyone
```

NFS Manager zur Konfiguration verwenden

Die Shareware NFS Manager von Marcel Bresink enthält eine graphische Benutzeroberfläche für die Erstellung von Einträgen in NetInfo `/exports`. Statt die Einträge manuell über den NetInfo-Manager oder die entsprechenden Kommandozeilenbefehle eingeben zu müssen, kann der Administrator

Abb. 3.57. NFS Manager verfügt über eine GUI für die Erstellung von Einträgen in NetInfo /exports.

die geforderten Attribute und Optionen komfortabel über einen graphischen Dialog definieren (Abb. 3.57).

Das Programm kann unter

`http://www.bresink.com/osx/NFSManager-de.html`

direkt von der Webseite des Entwicklers bezogen werden und ist für Testzwecke uneingeschränkt nutzbar. Für eine Seriennummer zur dauerhaften Nutzung verlangt der Autor eine Registrierungsgebühr von derzeit 17,40 EUR.

3.6.4 FTP-Server aktivieren

Natürlich enthält Mac OS X auch einen vollwertigen FTP-Server. Ähnlich wie der AFP-Server und der SMB-Server ist auch der FTP-Server bereits vorkonfiguriert und muss lediglich in den Systemeinstellungen Sharing aktiviert werden (Abb. 3.58).

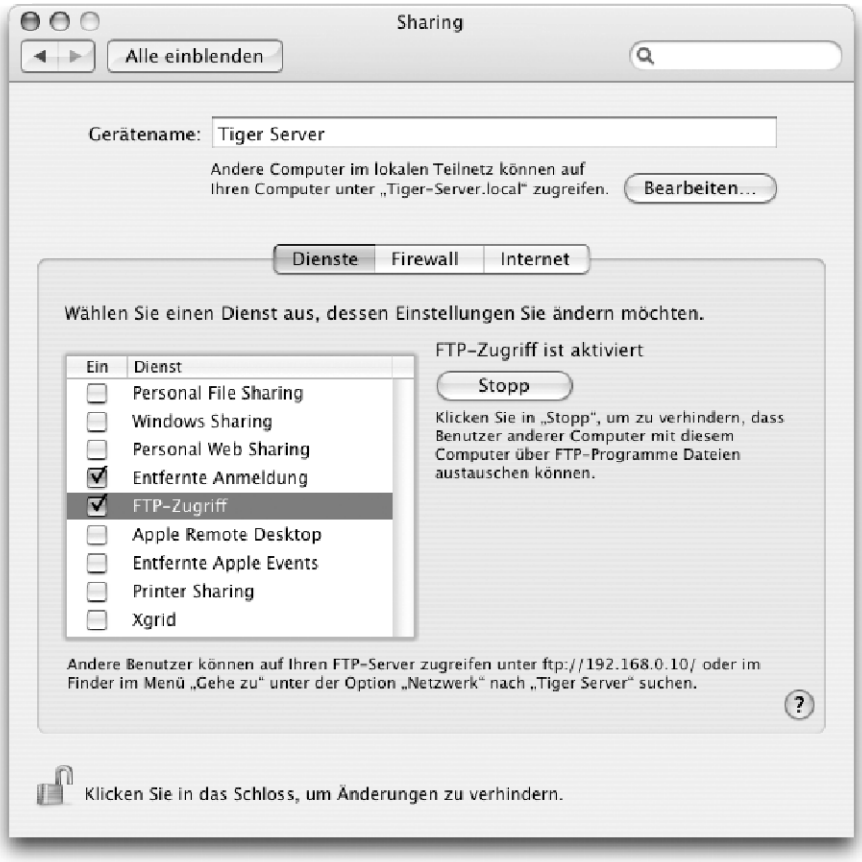


Abb. 3.58. Der FTP-Server ist bereits vorkonfiguriert und muss lediglich in den Systemeinstellungen Sharing aktiviert werden.

Der FTP-Server verlangt zur Authentifizierung den Namen und das Passwort eines lokalen Benutzers. Verläuft der Authentifizierungsprozess erfolgreich, kann der Benutzer auf alle lokalen Dateien zugreifen, auf die er bei einem lokalen Login auch Zugriff hätte. Insbesondere kann der Benutzer natürlich auf die Inhalte seines eigenen Home-Verzeichnisses lesend und schreibend zugreifen (nach einem erfolgreichen Login wechselt der FTP-Client automatisch in das Home-Verzeichnis des angemeldeten Benutzers), aber auch alle anderen Bereiche lesen, also z. B. die Verzeichnisse `/Applications`, `/Library` oder auch das im Finder nicht sichtbare Verzeichnis `/etc`:

```
$ ftp Tiger-Server.local
Trying fe80:4::203:93ff:fe19:696a...
Connected to tiger-server.local.
220 tiger-server.local FTP server (tnftpd 20040810) ready.
Name (Tiger-Server.local:rkk): admin
331 Password required for admin.
Password:
230-
    Welcome to Darwin!
230 User admin logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/Users/admin" is the current directory.
ftp> quit
221-
    Data traffic for this session was 0 bytes in 0 files.
    Total traffic for this session was 514 bytes in 0 transfers.
221 Thank you for using the FTP service on tiger-server.local.
```

Die Aktivierung des FTP-Servers erfolgt über `launchd`, d. h. auch der FTP-Serverprozess wird nur bei Bedarf gestartet:

```
$ cat /System/Library/LaunchDaemons/ftp.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
            "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>com.apple.ftpd</string>
        <key>Program</key>
        <string>/usr/libexec/ftpd</string>
        <key>ProgramArguments</key>
        <array>
                <string>ftpd</string>
                <string>-l</string>
        </array>
        <key>Sockets</key>
        <dict>
                <key>Listeners</key>
                <dict>
                        <key>Bonjour</key>
                        <true/>
                        <key>SockServiceName</key>
                        <string>ftp</string>
```

```
                </dict>
        </dict>
        <key>inetdCompatibility</key>
        <dict>
                <key>Wait</key>
                <false/>
        </dict>
</dict>
</plist>
```

Gast-Zugriff aktivieren

Der Gastzugriff – also ein Zugriff ohne eine gültige Benutzerkennung – ist im Auslieferungszustand nicht möglich:

```
$ ftp Tiger-Server.local
Trying fe80:4::203:93ff:fe19:696a...
Connected to tiger-server.local.
220 tiger-server.local FTP server (tnftpd 20040810) ready.
Name (Tiger-Server.local:rkk): anonymous
530 User anonymous unknown.
ftp: Login failed.
ftp>
...
```

Der FTP-Server interpretiert die Benutzernamen `anonymous` oder `ftp` als Gastkennungen. Falls auf dem Server-Rechner die Benutzerkennung `ftp` existiert, gewährt er in beiden Fällen (eingeschränkten) Zugriff mit einem beliebigen Passwort. Sofern der FTP-Server nicht explizit anders konfiguriert wird, erhält der Gastbenutzer Zugriff auf das Home-Verzeichnis des Benutzers `ftp`.

Auf Mac OS X fehlt die Kennung `ftp` und muss erstellt werden, um den Gastzugang für FTP zu aktivieren. Da es sich bei dieser Kennung um eine Systemkennung mit besonderen Eigenschaften handelt, sollte man sie nicht in den Systemeinstellungen Benutzer, sondern direkt in der NetInfo-Datenbank anlegen.

Am einfachsten geht es, wenn man im NetInfo-Manager eine der unprivilegierten Kennungen, wie beispielsweise `unknown` oder `nobody`, dupliziert. Auf diese Weise erhält man einen Eintrag, der bereits alle notwendigen Attribute enthält und nur noch geringfügig angepasst werden muss (Abb. 3.59).

Die Werte der Attribute `passwd` (kein Passwort, Login nicht möglich), `expire` (Passwort abgelaufen), `gid` (Gruppe `unknown`[37]), `shell` (keine Shell) und `change` (Passwort muss geändert werden) können unverändert bleiben.

Neben dem Namen (`name`) muss auch die Benutzernummer (`uid`) und das Home-Verzeichnis (`home`) modifiziert werden. Das Attribut `realname` ist weniger wichtig, sollte aber der Vollständigkeit halber ebenfalls geändert werden. Das Attribut `_writers_passwd` kann gelöscht werden, da wir keine Passwortänderungen durch den FTP-Gastbenutzer vorsehen wollen.

[37] Natürlich könnte man auch eine eigene FTP-Gruppe definieren.

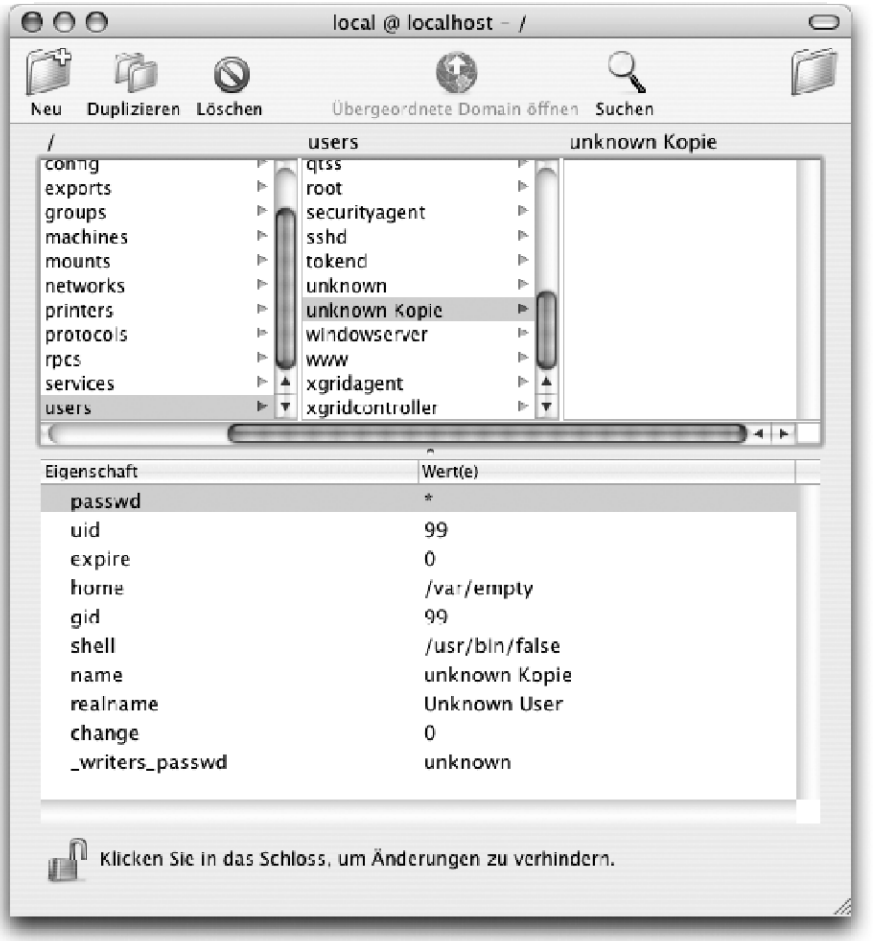


Abb. 3.59. Der FTP-Gastbenutzer lässt sich am einfachsten als Duplikat eines bereits vorhandenen, unprivilegierten Benutzers (hier „unknown“) anlegen.

Das Attribut `name` bezeichnet den Kurznamen bzw. das Login des Benutzers – der Wert von `name` muss daher in `ftp` geändert werden.

Das Attribut `uid` muss als Wert eine eindeutige Benutzernummer enthalten – vor der Änderung sollte man daher sicherstellen, dass die neue Benutzernummer noch nicht vergeben ist. Um zu verhindern, dass der Benutzername in der Benutzerliste des Login-Fensters auftaucht, sollte man dabei eine Benutzernummer wählen, die kleiner als 500 ist[38].

[38] Die erste, vom Mac-OS-X-Installationsprogramm angelegte (Administrator-) Benutzerkennung hat die Benutzernummer 501. Die Benutzernummern aller später mit Hilfe der Systemeinstellungen Benutzer angelegten Benutzerkennungen werden von Mac OS X einfach ab Nummer 501 fortlaufend vergeben.

Im Auslieferungszustand von Mac OS X 10.4 ist beispielsweise die Nummer 98 nicht vergeben und könnte daher als Benutzernummer für die Kennung `ftp` verwendet werden:

```
$ nireport / /users uid name | sort -n
-2      nobody
0       root
1       daemon
26      lp
27      postfix
70      www
71      eppc
74      mysql
75      sshd
76      qtss
77      cyrusimap
78      mailman
79      appserver
82      clamav
83      amavisd
84      jabber
85      xgridcontroller
86      xgridagent
87      appowner
88      windowserver
91      tokend
92      securityagent
99      unknown
501     admin
```

Das Attribut `home` muss den Pfad zum Home-Verzeichnis des Benutzers enthalten. Dieses Home-Verzeichnis ist im Falle des FTP-Gastbenutzers genau jenes Verzeichnis, auf das FTP-Gäste Zugriff erhalten sollen. Man kann ein Verzeichnis `/Users/ftp` anlegen und den Pfad dazu hier eintragen. Man kann aber natürlich ein beliebiges, anderes Verzeichnis, z. B. auf einer externen Festplatte (der Pfad beginnt dann mit `/Volumes/`) verwenden.

Den Wert des Attributs `realname` („Langer“ Name des Benutzers, z. B. „FTP User“) kann schließlich beliebig geändert werden.

Abbildung 3.60 zeigt einen auf diese Weise angelegten FTP-Gastbenutzer-Eintrag in der NetInfo-Datenbank.

Sobald ein solcher Eintrag erzeugt worden ist, können bei aktiviertem FTP-Server unter Angabe von `anonymous` oder `ftp` als Benutzername Gastverbindungen ohne Passwort aufgebaut werden:

```
$ ftp Tiger-Server.local
Trying fe80:4::203:93ff:fe19:696a...
Connected to tiger-server.local.
220 tiger-server.local FTP server (tnftpd 20040810) ready.
Name (Tiger-Server.local:rkk): anonymous
331 Guest login ok, type your name as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
```

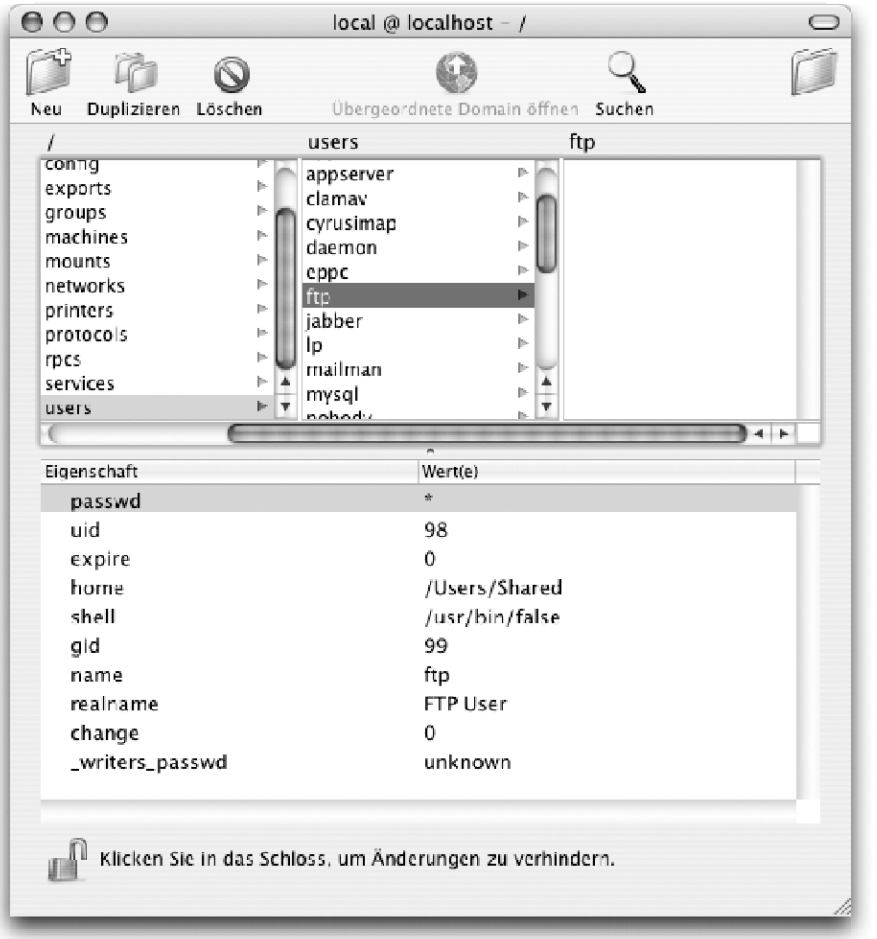


Abb. 3.60. Der FTP-Gastbenutzer als unprivilegierter Benutzer in der NetInfo-Datenbank.

```
257 "/" is the current directory.
ftp>
...
```

Der FTP-Gastbenutzer darf im Gegensatz zu authentifizierten Benutzern sein Home-Verzeichnis zwar nicht verlassen, darf in diesem jedoch entsprechend seiner Zugriffsrechte Dateien lesen und schreiben. Entsprechend genau sollte man auf die Zugriffsrechte dieses Verzeichnisses achten.

Da man (anonymes) FTP häufig als reines Veröffentlichungsmedium verwendet, kann es sinnvoll sein, dem FTP-Gastbenutzer den Schreibzugriff auf sein eigenes Home-Verzeichnis zu entziehen.

Am sichersten lässt sich das bewerkstelligen, indem man als root ein völlig neues Verzeichnis erzeugt und damit root zum Eigentümer dieses Verzeichnisses macht:

```
$ $ sudo mkdir ftp
$ ls -ld ftp
drwxr-xr-x  2 root  admin  68 26 Aug 13:23 ftp
```

Auf diese Weise darf ausschließlich root (und sonst niemand) in dieses Verzeichnis schreiben. Zu veröffentlichende Dateien legt man dann am besten in einem Unterverzeichnis ab – als Verzeichnisname für allgemein zugängliche (Download-)Verzeichnisse auf FTP-Servern hat sich der Name `pub` eingebürgert. Auch dieses Verzeichnis sollte ruhig root gehören. Allerdings ist es praktisch, einer Gruppe Schreibrechte zu geben, damit (lokale) Mitglieder dieser Gruppe Dateien in diesem Verzeichnis ablegen können. Das kann, muss aber nicht, einfach die Gruppe admin sein:

```
$ sudo mkdir ftp/pub
$ ls -ld ftp/pub
drwxr-xr-x  2 root  admin  68 26 Aug 13:28 ftp/pub
$ sudo chmod g+w ftp/pub
$ ls -ld ftp/pub
drwxrwxr-x  2 root  admin  68 26 Aug 13:28 ftp/pub
```

Möchte man nicht-authentifizierten FTP-Benutzern die Möglichkeit geben, Dateien nicht nur abzurufen, sondern auch abzulegen, kann man dafür ein weiteres Unterverzeichnis einrichten. Als Verzeichnisname für so ein Upload-Verzeichnis hat sich der Name `incoming` eingebürgert. Anders als beim Unterverzeichnis `pub` sollte man hier jedoch den Benutzer `ftp` zum Eigentümer von `incoming` machen, um FTP-Gästen effektiv den Schreibzugriff gewähren zu können. Um zu verhindern, dass (andere) FTP-Gäste den Inhalt dieses Verzeichnisses lesen können, kann man dem Eigentümer und dem Rest der Welt die Leserechte entziehen[39]. Einzig eine lokale Gruppe (z. B. admin) sollte sowohl Lese- als auch Schreibrechte für dieses Verzeichnis bekommen, um die Uploads bearbeiten zu können:

```
$ sudo mkdir ftp/incoming
$ ls -ld ftp/incoming
drwxr-xr-x  2 root  admin  68 26 Aug 15:03 ftp/incoming
$ sudo chown ftp ftp/incoming
$ ls -ld ftp/incoming
drwxr-xr-x  2 ftp  admin  68 26 Aug 15:03 ftp/incoming
$ sudo chmod u-r,g+w,o= ftp/incoming
$ ls -ld ftp/incoming
d-wxrwx---  2 ftp  admin  68 26 Aug 15:03 ftp/incoming
```

Eine weitere Konfiguration des FTP-Servers ist nicht notwendig. Die von FTP-Gästen angelegten Dateien sind in der Standardkonfiguration ausschließlich für Mitglieder der Gruppe lesbar, die dem Verzeichnis zugewiesen

[39] Andernfalls ist es heutzutage nur eine Frage der Zeit, bis der FTP-Server als Ablageplatz für (meist illegale) fremde Inhalte entdeckt wird.

wurde. Demnach können FTP-Gäste die abgelegten Dateien nach dem Ablegen selbst dann nicht mehr lesen, wenn sie den Dateinamen kennen – ein wirksames Mittel um Missbrauch des FTP-Servers zu verhindern. Zusätzlich verhindert der FTP-Server auch ohne weitere Konfiguration automatisch, dass einmal abgelegte Dateien durch später hochgeladene Dateien gleichen Namens überschrieben werden – was natürlich gerade für Verzeichnisse, deren Inhalt von den Benutzern nicht gelistet werden kann, besonders wichtig ist.

FTP-Zugriffe protokollieren

Da der FTP-Sever `ftpd` mit der Option `-l` startet, ist die Protokollierung normalerweise immer aktiv. Die Protokollierung erfolgt über `syslogd`. Die Protokolleinträge werden dabei in die Datei `/var/log/ftp.log` geschrieben. Protokolliert werden normalerweise lediglich die An- und Abmeldevorgänge:

```
$ cat /var/log/ftp.log
...
Aug 26 15:22:38 localhost ftpd[1955]: FTP LOGIN FROM localhost as
                                      rkk (class: real, type: REAL)
Aug 26 15:22:42 localhost ftpd[1955]: Data traffic: 0 bytes in
                                      0 files
Aug 26 15:22:42 localhost ftpd[1955]: Total traffic: 565 bytes in
                                      0 transfers
Aug 26 15:33:43 localhost ftpd[1990]: connection from localhost to
                                      localhost
...
Aug 26 15:48:16 localhost ftpd[2203]: connection from localhost to
                                      localhost
Aug 26 15:48:21 localhost ftpd[2203]: ANONYMOUS FTP LOGIN FROM
                                      localhost,  (class: guest,
                                      type: GUEST)
Aug 26 15:48:29 localhost ftpd[2203]: Data traffic: 0 bytes in
                                      0 files
Aug 26 15:48:29 localhost ftpd[2203]: Total traffic: 559 bytes in
                                      0 transfers
...
```

Wird die Option `-l` jedoch zweimal angegeben, lassen sich zusätzlich die wichtigsten Datei- und Verzeichnisoperationen zusammen mit ihren Argumenten mitprotokollieren, d. h. Datei lesen (get), Datei schreiben (put), Datei ergänzen (append), Datei löschen (delete), Verzeichnis anlegen (mkdir), Verzeichnis löschen (rmdir) und Objekt umbenennen (rename):

```
...
Aug 26 16:01:43 localhost ftpd[2221]: connection from localhost to
                                      localhost
Aug 26 16:01:46 localhost ftpd[2221]: ANONYMOUS FTP LOGIN FROM
                                      localhost,  (class: guest,
                                      type: GUEST)
Aug 26 16:01:55 localhost ftpd[2221]: get /pub/README = 21 bytes in
                                      0.000 seconds
Aug 26 16:01:56 localhost ftpd[2221]: Data traffic: 21 bytes in
                                      1 file
Aug 26 16:01:56 localhost ftpd[2221]: Total traffic: 1248 bytes
                                      in 3 transfers
...
```

3.6.5 HTTP/WebDAV-Server aktivieren

Mac OS X verfügt mit Apache über einen besonders leistungsfähigen und vielseitigen HTTP-Server. Apache ist open-source und für nahezu alle gängigen Betriebssysteme verfügbar. Der Apache HTTP-Server ist mit großem Abstand der derzeit am weitesten verbreitete HTTP-Server: aktuelle Untersuchungen bescheinigen Apache einen Marktanteil von mehr als zwei Dritteln![40].

Bei dem mit Mac OS X 10.4.2 ausgelieferten Apache handelt es sich um die Version 1.3.33:

```
$ httpd -v
Server version: Apache/1.3.33 (Darwin)
Server built:   Mar 20 2005 15:08:27
```

Apache kann am einfachsten über die Systemeinstellungen Sharing aktiviert werden. Ähnlich wie auch bei den anderen Serverdiensten, hat Apple dort auch für den HTTP-Server keine Konfigurationsmöglichkeiten vorgesehen – als „Personal Web Sharing“ kann der Apache HTTP-Server dort lediglich gestartet und gestoppt werden (Abb. 3.61).

Damit der HTTP-Server nach der Aktivierung in den Systemeinstellungen Sharing auch nach dem nächsten Neustart des Systems automatisch gestartet wird, vermerkt Mac OS X den Status von „Personal Web Sharing“ in der Datei `/etc/hostconfig`.

```
$ cat /etc/hostconfig
...
WEBSERVER=-YES-
...
```

Der Wert der Variablen `WEBSERVER` wird beim Start im Startobjekt Apache abgefragt. Lautet der Wert `-YES-`, wird dort der HTTP-Server mit Hilfe des Befehls `apachectl` gestartet:

```
$ cat /System/Library/StartupItems/Apache/Apache
...
StartService ()
{
    if [ "${WEBSERVER:=-NO-}" = "-YES-" ]; then
        echo "Starting Apache web server"
        if [ ! -e /etc/httpd/httpd.conf ] ; then
                cp -p /etc/httpd/httpd.conf.default
                      /etc/httpd/httpd.conf
        fi
        apachectl start
        if [ "${WEBPERFCACHESERVER:=-NO-}" = "-YES-" ]; then
            if [ -x /usr/sbin/webperfcachectl ]; then
                echo "Starting web performance cache server"
                /usr/sbin/webperfcachectl start
            fi
```

[40] http://news.netcraft.com/archives/2004/08/01/august_2004_web_server_survey.html

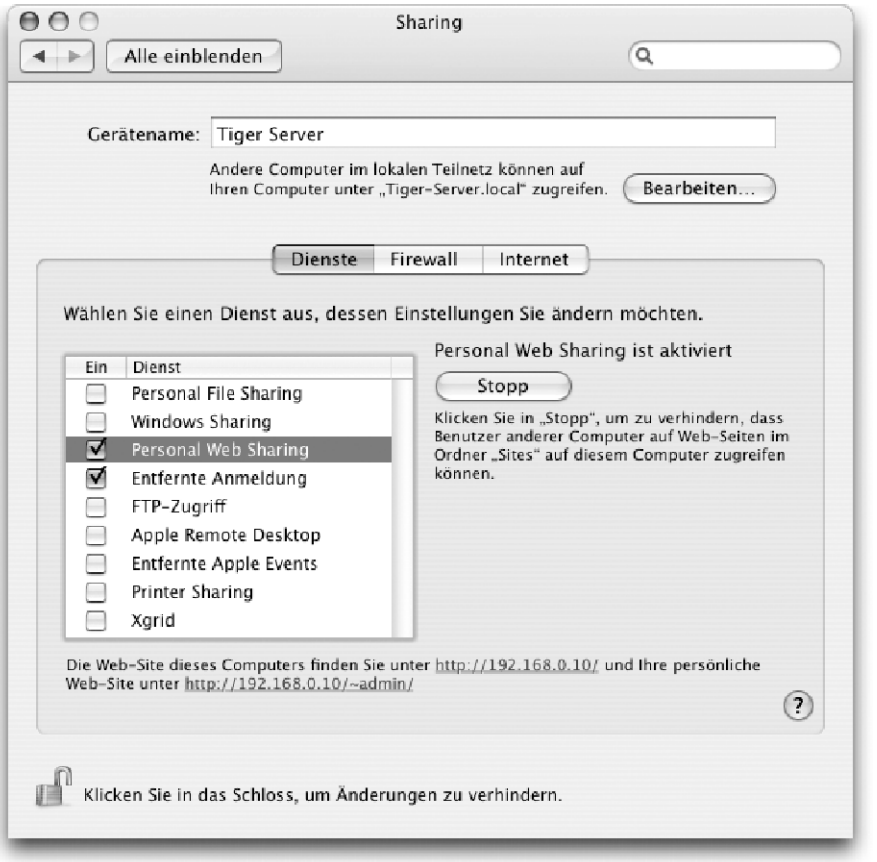


Abb. 3.61. Der Apache-HTTP-Server kann in den Systemeinstellungen Sharing als „Personal Web Sharing“ gestartet werden.

```
        fi
    fi
}
...
```

Sollte die Konfigurationsdatei des HTTP-Servers `/etc/httpd/httpd.conf` fehlen, stellt sie das Skript mit dem Befehl `cp` als Kopie der Standardkonfigurationsdatei `httpd.conf.default` wieder her – praktisch zu wissen, wenn man nach eigenen (fehlgeschlagenen) Konfigurationsexperimenten mal den Wunsch nach dem Zurücksetzen der Konfiguration verspürt.

Mit `apachectl` lässt sich der HTTP-Server natürlich auch direkt in der Befehlszeile starten (`apachectl start`) – und stoppen (`apachectl stop`):

```
$ sudo apachectl start
Processing config directory: /private/etc/httpd/users/*.conf
 Processing config file: /private/etc/httpd/users/admin.conf
```

```
 Processing config file: /private/etc/httpd/users/rkk.conf
 ...
/usr/sbin/apachectl start: httpd started
$ sudo apachectl stop
/usr/sbin/apachectl stop: httpd stopped
```

Nach dem Start bedient der HTTP-Server Anfragen von Clients mit Dokumenten aus dem Verzeichnis `/Library/WebServer/Documents/` und aus den Unterverzeichnissen `Sites` (im deutschen Finder `Web-Sites`), die sich in den Home-Verzeichnissen der lokalen Benutzer befinden.

Der Inhalt von `/Library/WebServer/Documents/` ist als *DocumentRoot* konfiguriert, so dass zum Abruf von Dokumenten aus diesem Verzeichnis die Angabe des Rechnernamens (ohne Pfad) in der URL genügt.

Abb. 3.62. Das Verzeichnis /Library/WebServer/Documents enthält im Auslieferungszustand eine Testseite mit einem Link auf die lokal installierte Apache-Online-Dokumentation.

Dieses Verzeichnis ist mit einer Demonstrationsseite (Abb. 3.62) vorkonfiguriert, die einen Link auf die lokal installierte Apache-Online-Dokumentation enthält. Um diese Demonstrationsseite, bzw. die Online-Dokumentation aufzurufen, muss man lediglich eine der folgenden URLs eingeben:

```
http://localhost
http://localhost/manual
```

Um die von den einzelnen Benutzern in ihren Home-Verzeichnissen (Unterverzeichnis `Sites`) abgelegten persönlichen HTML-Dokumente abzurufen, muss man die URL um ein Tilde-Zeichen (`~`) und um den Kurznamen des Benutzers ergänzen, z. B.:

```
http://localhost/~admin
http://localhost/~rkk
```

Im Auslieferungszustand bekommt man eine einfache Testseite zu sehen (Abb. 3.63).

Dem Benutzer stehen in Mac OS X demnach normalerweise zwei Verzeichnisse zur Verfügung, die er für die Veröffentlichung von Dokumenten mittels HTTP verwenden kann. Das Verzeichnis `/Library/WebServer/Documents/`

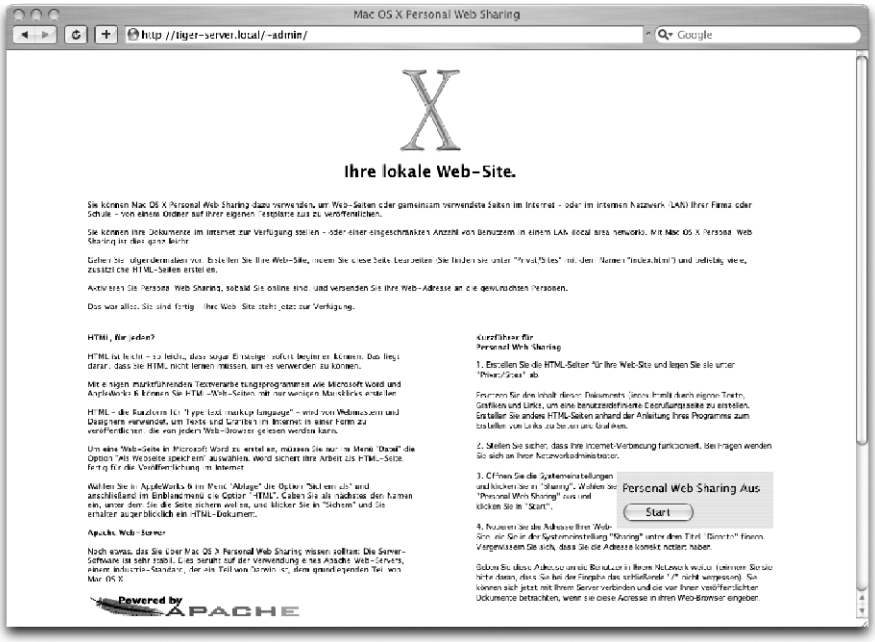


Abb. 3.63. Jeder lokale Benutzer verfügt in seinem Home-Verzeichnis über ein Unterverzeichnis Sites, das zum Veröffentlichen von persönlichen Informationen per HTTP verwendet werden kann. Im Auslieferungszustand enthält dieses Verzeichnis eine einfache Testseite.

ist dabei das designierte, systemweite Verzeichnis des HTTP-Servers, auf das nur Administratoren Schreibzugriff erhalten. Für den Inhalt des persönlichen Verzeichnisses `~/Sites/` ist dagegen jeder einzelne Benutzer selbst verantwortlich.

Apache konfigurieren

Die Apache-Konfigurationsdateien liegen im Verzeichnis `/etc/httpd/`. Nach dem Start liest Apache als erstes die in der Hauptkonfigurationsdatei `httpd.conf` enthaltenen Anweisungen. Durch Änderungen an dieser Datei kann das Verhalten des HTTP-Servers im Detail an die eigenen Bedürfnisse und Wünsche angepasst werden.

Die in dieser Konfigurationsdatei enthaltenen Anweisungen lassen sich ganz grob in globale und verzeichnisspezifische Anweisungen unterteilen. Die globalen Konfigurationsanweisungen legen Parameter fest, die für den HTTP-Server als Ganzes gelten sollen, während in den verzeichnisspezifischen Abschnitten die Eigenschaften bestimmter (Unter-)Verzeichnisse definiert werden.

Zu den globalen Anweisungen gehört z. B. auch die Festlegung der designierten, systemweiten Freigabe:

```
$ cat /etc/httpd/httpd.conf
...
#
# DocumentRoot: The directory out of which you will serve
# your documents. By default, all requests are taken from
# this directory, but symbolic links and aliases may be used
# to point to other locations.
#
DocumentRoot "/Library/WebServer/Documents"
...
```

Die mit einem #-Zeichen beginnenden Zeilen werden als Kommentare ignoriert.

Die Eigenschaften von `/Library/WebServer/Documents/` sind dann etwas weiter unten in der gleichen Datei definiert:

```
...
#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/Library/WebServer/Documents">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly*~--~"Options
# All" doesn't give it to you.
#
     Options Indexes FollowSymLinks MultiViews
```

```
#
# This controls which options the .htaccess files in directories
# can override. Can also be "All", or any combination of "Options",
# "FileInfo", "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all
</Directory>
...
```

Diese Anweisungen legen fest, dass

- ein Verzeichnisindex automatisch erstellt wird, falls die Standard-Index-Datei – meistens index.html – fehlen sollte (`Options Indexes`),
- symbolische Links weiterverfolgt werden, auch wenn sie auf Ziele außerhalb dieses Verzeichnisses zeigen (`Options FollowSymLinks`),
- Dateianforderungen mit unterschiedlichen Dateien beantwortet werden können, je nachdem welche Sprache/Kodierung/Dateiformat der Client bevorzugt (`Options MultiViews`),
- *keine* dieser Anweisungen mit Hilfe von in den entsprechenden Verzeichnissen abgelegten `.htaccess`-Dateien außer Kraft gesetzt werden kann (`AllowOverride None`),
- und schließlich, dass alle Hosts unabhängig von deren DNS-Namen und IP-Adressen Zugriff erhalten sollen (`Order allow,deny` und `Allow from all`).

Zusätzliche Konfigurationsanweisungen für die benutzerspezifischen `Sites`-Verzeichnisse sind ausgelagert, und befinden sich in den von Mac OS X automatisch angelegten `*.conf`-Dateien im Verzeichnis `/etc/httpd/users/` – sie werden innerhalb von Haupkonfigurationsdatei `httpd.conf` mit Hilfe der `Include`-Anweisung referenziert und daher immer mitberücksichtigt:

```
$ cat /etc/httpd/httpd.conf
...
Include /private/etc/httpd/users/*.conf
```

Die dort getroffenen Festlegungen entsprechen nahezu vollständig den Festlegungen für `/Library/WebServer/Documents/`:

```
$ cat /etc/httpd/users/rkk.conf
<Directory "/Users/rkk/Sites/">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Der einzige Unterschied – neben dem Fehlen der ausführlichen Kommentare – besteht darin, dass symbolische Links, die aus den `Sites`-Verzeichnissen

hinausführen würden, nicht erlaubt sind (die Option `FollowSymLinks` fehlt). Auf diese Weise soll verhindert werden, dass ein nicht-privilegierter Benutzer beliebige, für ihn lesbare Bereiche des lokalen Dateisystems mit Hilfe eines symbolischen Links der (Web-)Öffentlichkeit preisgibt. Diese Freiheit soll in der Standardkonfiguration den Administratoren vorbehalten bleiben.

Damit im laufenden Betrieb etwaige Konfigurationsänderungen Wirkung zeigen, muss Apache seine Konfigurationsdateien neu einlesen. Der sicherste Weg hierzu ist der Neustart des HTTP-Servers über Systemeinstellungen Sharing oder über den Befehl `apachectl`:

```
$ sudo apachectl restart
/usr/sbin/apachectl restart: httpd restarted
```

In beiden Fällen werden alle aktiven Verbindungen von HTTP-Clients unterbrochen und die Änderungen damit sofort aktiv. Soll die geänderte Konfiguration hingegen nur für alle neuen HTTP-Verbindungen gelten, kann der HTTP-Server auch „sanft“ auf die geänderten Konfigurationsdaten hingewiesen werden:

```
$ sudo apachectl graceful
/usr/sbin/apachectl graceful: httpd gracefully restarted
```

Verzeichnisse zentral schützen

Normalerweise kann auf alle Dateien, die sich in einem der freigegebenen Verzeichnisse des HTTP-Servers befinden, ohne Authentifizierung zugegriffen werden: die Angabe einer URL, d. h. die Kenntnis des Dateinamens und des dazugehörigen Pfads genügt.

Einzelne Verzeichnisse (und deren Unterverzeichnisse) lassen sich jedoch verhältnismäßig einfach mit Hilfe eines Passworts schützen, sofern ein völlig freier Zugang unerwünscht ist. Gleichzeitig (oder auch alternativ) lässt sich der Zugriff auf Hosts mit bestimmten DNS-Namen bzw. IP-Adressen limitieren.

Um Verzeichnisse mit Hilfe von Passwörtern schützen zu können, muss als erstes eine Benutzerliste mit Namen und Passwörtern für zugriffsberechtigte Benutzer angelegt werden[41].

Apache unterstützt von Haus aus drei Arten von Benutzerlisten:

- Textdateien mit Benutzernamen und Crypt-Hashes der dazugehörigen Passwörter (*Basic Authentication*).
- Datenbank-Dateien mit Benutzernamen und Crypt-Hashes der dazugehörigen Passwörter (*Database Authentication*). Diese Art der Authentifizierung ist deutlich schneller als Basic Authentication und daher besonders für längere Benutzerlisten geeignet. Sie erfordert die Aktivierung des Apache-Moduls `mod_auth_dbm`.

[41] In der Einzelplatzvariante von Mac OS X fehlen die Mac-OS-X-Server spezifischen Apache-Module mod_auth_apple und mod_digest_apple, so dass Benutzer nicht über NetInfo bzw. Open Directory authentifiziert werden können und in jedem Fall eine separate Benutzerliste gepflegt werden muss.

- Textdateien mit Benutzernamen und MD5-Hashes der dazugehörigen Passwörter (*Digest Authentication*). Diese Art der Authentifizierung ist, da das Passwort des Benutzers nicht im Klartext übertragen wird, etwas sicherer als Basic und Database Authentication, aber leider nicht mit allen Web-Browsern kompatibel. Sie erfordert die Aktivierung des Apache-Moduls `mod_auth_digest`.

Das Basic-Authentication-Verfahren ist zwar aufgrund des im Klartext übertragenen Passworts recht unsicher, aber sehr verbreitet, da es von nahezu allen Web-Browsern unterstützt wird.

Um es zu aktivieren, muss im ersten Schritt eine Benutzerliste mit Zeilen im Format `benutzername:passwort` angelegt werden. Der Benutzername kann natürlich frei gewählt werden – beim Passwort muss es sich um einen Crypt-Hash des eigentlichen Benutzerpassworts handeln.

Diese Datei lässt sich am einfachsten mit dem Befehl `htpasswd` anlegen, der zum Apache-Paket gehört. `htpasswd` erwartet den Dateinamen der Benutzerliste und einen Benutzernamen als Parameter und fragt dann interaktiv das Passwort ab. Nach der Berechnung des Crypt-Hashes erzeugt der Befehl einen neuen Eintrag in der Benutzerliste. Die Option `-c` kann verwendet werden, um eine völlig neue Benutzerliste anzulegen:

```
$ cd /etc/httpd/
$ sudo htpasswd -c htpasswd rkk
New password:
Re-type new password:
Adding password for user rkk
$ cat htpasswd
rkk:nMEnFycaUSQAg
```

Im obigen Beispiel wurde eine Benutzerliste mit dem Dateinamen `htpasswd` im (Apache-)Verzeichnis `/etc/httpd/` zusammen mit einem Eintrag für den Benutzer `rkk` erzeugt (eine alte Benutzerliste gleichen Namens hätte der Befehl ohne Nachfrage überschrieben). Als Passwort wurde hier ebenfalls `rkk` verwendet, was leicht überprüft werden kann:

```
$ openssl passwd -salt nM rkk
nMEnFycaUSQAg
```

Da auf diese Weise gespeicherte Passwörter zwar nicht ohne weiteres entschlüsselt werden können, aber Neugierige nicht davon abhalten können, in kürzester Zeit eine große Menge von nahe liegenden Passwörtern einfach durchzuprobieren, sollte natürlich auf restriktive Zugriffsrechte für die Benutzerliste geachtet werden.

Auf jeden Fall sollte verhindert werden, dass jedermann die Datei lesen kann. Da der HTTP-Server selbst jedoch Leserechte benötigt, sollte man gleichzeitig die Gruppenzuordnung ändern: Apache muss zwar mit root-Rechten gestartet werden, läuft nach der ersten Initialisierung aber als Prozess des Benutzers `www` mit Primärgruppe `www` weiter. Um die Datei zu schützen, ohne den HTTP-Server zu behindern, muss man demnach dem

Rest der Welt die Leserechte entziehen und die Gruppe der Datei von `wheel` in `www` ändern:

```
$ ls -l htpasswd
-rw-r--r--  1 root  wheel     18 31 Aug 14:10 htpasswd
$ sudo chmod 640 htpasswd
$ sudo chown root:www htpasswd
$ ls -l htpasswd
-rw-r-----  1 root  www  18 31 Aug 14:10 htpasswd
```

Hat man die Benutzerliste einmal erstellt, kann man den Zugriff auf einzelne Verzeichnisse (und deren Unterverzeichnisse) von der Eingabe eines dort definierten Namens und Passworts abhängig machen.

Um eine komplette Freigabe zu schützen, können die dazu erforderlichen Anweisungen im verzeichnisspezifischen Bereich der Konfigurationsdatei eingefügt werden. Um beispielsweise die Freigabe `/Users/rkk/Sites/` zu schützen, müsste man die Konfigurationsdatei `/etc/httpd/users/rkk.conf` etwa in die folgende Form bringen:

```
$ cat /etc/httpd/users/rkk.conf
<Directory "/Users/rkk/Sites/">
    Options Indexes MultiViews
    AllowOverride None

    AuthName "Geheimprojekt"
    AuthType Basic
    AuthUserFile /etc/httpd/htpasswd
    Require user rkk

    Order allow,deny
    Allow from all
</Directory>
```

Gegenüber der ursprünglichen Fassung wurden hier die Befehle `AuthName`, `AuthType`, `AuthUserFile` und `Require` zusätzlich eingefügt. Der Befehl `AuthType` definiert dabei einfach die Art des Authentifizierungsverfahrens (hier Basic Authentication) und der Befehl `AuthUserFile` den Pfad zur zuvor erstellten Benutzerliste. Mit dem Befehl `Require` wird definiert, ob alle (`Require valid-user`) oder, wie im Beispiel, nur ausgewählte Benutzer Zugang zu dem geschützten Bereich erhalten sollen.

Der Befehl `AuthName` gibt diesem geschützten Bereich, der manchmal auch als *Sicherheitszone* oder englischsprachig als *Realm* bezeichnet wird, einen eindeutigen Namen. Aufgrund dieses Namens lassen sich mehrere geschützte Bereiche auf einem einzigen HTTP-Server unterscheiden.

Der HTTP-Server beantwortet nicht-authentifizierte Anfragen nach Dokumenten aus einem geschützten Bereich grundsätzlich mit einer Fehlermeldung. Diese Fehlermeldung fordert zur Authentifizierung auf und enthält immer auch den Namen des geschützten Bereichs. Ist dieser Name bisher unbekannt, bittet der HTTP-Client (Web-Browser) den Benutzer um die Eingabe eines Namens und eines Passworts (Abb. 3.64).

Da HTTP zustandslos arbeitet, wiederholt sich dieser Ablauf mit jeder weiteren Anfrage. Damit der Benutzer jedoch nicht jedes Mal sein Passwort

Abb. 3.64. Der HTTP-Server beantwortet nicht-authentifizierte Anfragen nach Dokumenten aus einem geschützten Bereich grundsätzlich mit einer Fehlermeldung, die zur Authentifizierung auffordert und den Namen des geschützten Bereichs enthält.

neu eingeben muss, speichert der HTTP-Client den eingegebenen Benutzernamen und das Passwort zusammen mit dem Namen des geschützten Bereichs. Auf diese Weise kann der HTTP-Client wiederholte Anfragen an denselben geschützten Bereich auch ohne Interaktion mit dem Benutzer absetzen.

Nach Änderung der Konfigurationsdatei muss Apache nur noch neu gestartet werden, um den Schutz zu aktivieren.

Verzeichnisse dezentral schützen

Um die Apache-Konfigurationsdateien zu ändern, sind unter Mac OS X mindestens Administrator-Rechte erforderlich. Um nicht-privilegierten Benutzern mehr Kontrolle über ihre eigenen `Sites`-Verzeichnisse zu geben, kann man sie aber auch selbst entscheiden lassen, wem sie Zugriff auf ihre dort abgelegten Dateien gewähren wollen und wem nicht.

Verzeichnisspezifische Konfigurationsanweisungen werden nämlich nicht ausschließlich aus der Hauptkonfigurationsdatei gelesen, sondern können auch dezentral direkt in den freigegebenen Verzeichnissen abgelegt werden.

Der einheitliche Dateiname – normalerweise `.htaccess` – dieser dezentralen Konfigurationsdateien wird in der Hauptkonfigurationsdatei von Apache definiert:

```
$ cat /etc/httpd/httpd.conf
...
```

```
#
# AccessFileName: The name of the file to look for in each
# directory for access control information.
#
AccessFileName .htaccess
...
```

Um ein Verzeichnis mit Hilfe einer solchen Datei zu schützen, legt man eine Datei dieses Namens darin ab und schreibt dorthin die Anweisungen, die man sonst in den verzeichnisspezifischen Bereich der Konfigurationsdatei geschrieben hätte[42], z. B.

```
$ cat /Users/rkk/Sites/.htaccess
AuthName "Geheimprojekt"
AuthType Basic
AuthUserFile /etc/httpd/htpasswd
Require user rkk
```

Abhängig vom Parameter des verzeichnisspezifischen Befehls `AllowOverride` berücksichtigt oder ignoriert Apache die in solchen Dateien abgelegten Konfigurationsbefehle. In der Standardeinstellung werden `.htaccess` Dateien aus Sicherheitsgründen ignoriert. Um sie zu berücksichtigen, muss der Administrator einmalig die verzeichnisspezifischen Einstellungen ändern und entweder mit `AllowOverride All` die Änderung aller verzeichnisbezogenen Einstellungen, oder mit `AllowOverride AuthConfig` die Änderung der Authentifizierungseinstellungen erlauben.

Damit die in der oben dargestellten Beispieldatei `/Users/rkk/Sites/.htaccess` enthaltenen Befehle berücksichtigt werden, müsste man demnach in der Konfigurationsdatei `/etc/httpd/users/rkk.conf` den Befehl `AllowOverride None` durch `AllowOverride AuthConfig` ersetzen:

```
$ cat /etc/httpd/users/rkk.conf
<Directory "/Users/rkk/Sites/">
    Options Indexes MultiViews
    AllowOverride AuthConfig
    Order allow,deny
    Allow from all
</Directory>
```

Sobald die dezentrale Konfiguration einmal freigeschaltet ist, sind Änderungen der Konfiguration ohne Neustart des HTTP-Servers möglich. Da die `.htaccess`-Dateien bei jedem Zugriff auf ein Verzeichnis gelesen werden, sind Konfigurationsänderungen entsprechend sofort wirksam.

Außer von Apache können die `.htaccess`-Dateien im Übrigen höchstens von anderen lokalen Benutzern gelesen werden. In der Hauptkonfigurationsdatei ist festgelegt, dass diese Dateien (und alle anderen, die mit der Zeichenfolge `.ht` beginnen) grundsätzlich vom Zugriff über HTTP ausgeschlossen sind:

```
$ cat /etc/httpd/httpd.conf
...
```

[42] Es steht jedem Benutzer natürlich frei, eine eigene Benutzerliste in seinem eigenen Home-Verzeichnis anzulegen und zu pflegen.

```
#
# The following lines prevent .htaccess files from being viewed by
# Web clients.  Since .htaccess files often contain authorization
# information, access is disallowed for security reasons.  Comment
# these lines out if you want Web visitors to see the contents of
# .htaccess files.  If you change the AccessFileName directive
# above, be sure to make the corresponding changes here.
#
# Also, folks tend to use names such as .htpasswd for password
# files, so this will protect those as well.
#
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>
...
```

Schreibzugriff mit WebDAV erlauben

Normalerweise kann man auf HTTP-Freigaben ausschließlich lesend zugreifen. Mit der HTTP-Erweiterung WebDAV lässt sich aber für bestimmte Verzeichnisse auch der Schreibzugriff realisieren. Man muss die WebDAV-Unterstützung in Apache dazu allerdings erst explizit aktivieren, da sie im Auslieferungszustand von Mac OS X deaktiviert ist.

Die WebDAV-Unterstützung ist kein integraler Bestandteil von Apache, sondern muss, wie viele andere Eigenschaften von Apache, bei Bedarf zur Laufzeit (d. h. dynamisch) als Apache-Modul geladen werden. Die für das Laden und Aktivieren des WebDAV-Moduls relevanten Anweisungen sind in der Konfigurationsdatei `/etc/httpd/httpd.conf` bereits vorhanden und lediglich auskommentiert. Um das Modul zu laden und zu aktivieren, muss man also lediglich die folgenden Zeilen finden und die Kommentarzeichen am Zeilenanfang entfernen:

```
$ cat /etc/httpd/httpd.conf
...
#LoadModule dav_module         libexec/httpd/libdav.so
...
#AddModule mod_dav.c
...
```

Das WebDAV-Modul erfordert nur minimale Konfiguration. Um einen Speicherort und einen Timeout für Dateizugriffssperren zu definieren, fügt man in der Apache-Hauptkonfigurationsdatei einfach die folgenden Anweisungen ein:

```
$ cat /etc/httpd/httpd.conf
...
<IfModule mod_dav.c>
        DAVLockDB "/var/run/davlocks/davlockdb"
        DAVMinTimeout 600
</IfModule>
...
```

Um WebDAV für eine bestimmte HTTP-Freigabe zu aktivieren, muss man innerhalb des entsprechenden `Directory`-Bereichs die Anweisungen `DAV On` einfügen. In den meisten Fällen wird man dabei den WebDAV-Zugriff zusammen mit Authentifizierung verwenden wollen. Im einfachsten Fall genügt es, einfach WebDAV und Authentifizierung zu aktivieren:

```
$ cat /etc/httpd/users/rkk.conf
<Directory "/Users/rkk/Sites/Geheimprojekt/">
    Options Indexes MultiViews
    AllowOverride None

    DAV On

    AuthName "Geheimprojekt"
    AuthType Basic
    AuthUserFile /etc/httpd/htpasswd
    Require user rkk

    Order allow,deny
    Allow from all
</Directory>
```

Eine so veränderte Konfigurationsdatei für das Verzeichnis `/Users/rkk` erlaubt sowohl den Schreib- als auch den Lesezugriff ausschließlich den in der `Require`-Anweisung genannten Benutzern.

Alternativ ist allerdings auch die etwas weniger restriktive nachfolgend gelistete Konfiguration möglich, die lediglich den Schreib- aber nicht den Lesezugriff einschränkt:

Abb. 3.65. Entsprechend konfiguriert, erlaubt Apache Schreibzugriffe mit WebDAV nur korrekt authentifizierten Benutzern.

```
$ cat /etc/httpd/users/rkk.conf
<Directory "/Users/rkk/Sites/Geheimprojekt/">
    Options Indexes MultiViews
    AllowOverride None

    DAV On

    AuthName "Geheimprojekt"
        AuthType Basic
    AuthUserFile /etc/httpd/htpasswd

    <Limit PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY
                                          MOVE LOCK UNLOCK>
      Require user rkk
    </Limit>

    Order allow,deny
    Allow from all
</Directory>
```

Als Mittel zum Zweck dient hier die `Limit`-Anweisung, mit der sich die Granularität der Authentifizierung auf der Ebene der HTTP-Methoden festlegen lässt. Bei den im Beispiel gelisteten zehn Methoden (PUT, POST, usw.) handelt es sich genau um die gegenüber dem ursprünglichen HTTP-Standard hinzugefügten WebDAV-Methoden. Mit Hilfe dieser `Limit`-Anweisung wird demnach effektiv festgelegt, dass WebDAV-Zugriffe (also Schreibzugriffe) authentifiziert und reguläre HTTP-Zugriffe (also Lesezugriffe) nicht authentifiziert werden müssen.

Ein auf diese Weise geschütztes Verzeichnis kann ohne die Eingabe eines Passworts gelesen werden. Damit ist insbesondere der lesende Zugriff mit Hilfe eines Web-Browsers erlaubt. Schreibzugriffe mittels WebDAV müssen authentifiziert werden und werden nur demjenigen genehmigt, der sich mittels des richtigen Passworts erfolgreich als Benutzer `rkk` ausweisen kann (Abb. 3.65)

Alle mittels WebDAV erzeugten Dateien gehören dabei dem Benutzer `www`, weil sie lokal von Apache angelegt werden. Umgekehrt muss das freigegebene Verzeichnis für den Benutzer `www` oder die Gruppe `www` schreibbar sein, damit Apache Dateien anlegen kann.

4

Verzeichnisdienste

Moderne Betriebssysteme benötigen eine große Menge von Konfigurationsdaten (Benutzer, Gruppen, Netzwerkeinstellungen u. a.), die traditionell in diversen Konfigurationsdateien gespeichert werden. Auf Unix-basierten Systemen wurden die Benutzerdaten in der Vergangenheit z. B. häufig in der Datei `/etc/passwd` und die Gruppendaten in der Datei `/etc/groups` gespeichert.

Da heutzutage im professionellen Einsatz nur noch ganz selten mit einem einzelnen Rechner, sondern selbst in den kleinsten Unternehmen und Arbeitsgruppen mit einem halben Dutzend und mehr Rechnern gearbeitet wird, erwächst recht schnell der Wunsch nach gleichartigen Konfigurationen einer ganzen Gruppe von Rechnern.

In der Vergangenheit gab es hierfür lediglich zwei Möglichkeiten: entweder man übertrug die Konfigurationsdaten manuell auf alle zu konfigurierenden Rechner, oder man kopierte bei entsprechender Kenntnis des Betriebssystems direkt die relevanten Konfigurationsdateien.

Letzteres lässt sich mit Hilfe von geeigneten Werkzeugen (in Unix-basierten kam häufig der Befehl `rsync` zum Einsatz) natürlich automatisieren und regelmäßig (z. B. mit Hilfe von `cron`) ausführen.

Heutzutage versucht man dieses Problem komplett zu umgehen, indem die für die rechnerübergreifende Konfiguration relevanten Daten konsolidiert und in einer zentralen Datenbank ablegt werden. Dort können sie zentral gepflegt werden und sind für alle an diese zentrale Datenbank angeschlossenen Rechner zugänglich.

Eine solche zentrale Datenbank für Konfigurationsdaten wird als Verzeichnisdatenbank bezeichnet. Verzeichnisdienste stellen über ein definiertes Zugriffsprotokoll die Verbindung zwischen der Verzeichnisdatenbank und dem Betriebssystem her.

In Mac OS X werden alle Verzeichnisdienste von der Komponente *Open Directory* abstrahiert. Mit der Bezeichnung Open Directory wird zum Ausdruck gebracht, dass Mac OS X nicht auf eine bestimmte Art von Verzeichnis-

datenbanken festgelegt ist, sondern mit einer Vielzahl von Systemen zusammenarbeiten kann. Das Betriebssystem greift auf die Verzeichnisdienste nicht direkt zu, sondern bedient sich Open Directory als Dolmetscher, der die Anfragen des Betriebssystems in das spezifische Zugriffsprotokoll des jeweiligen Verzeichnisdienstes übersetzen kann.

Im Einzelnen unterstützt Open Directory in Mac OS X von Haus aus die folgenden Verzeichnisdienste:

- *LDAPv3*, ein offenes, herstellerunabhängiges Standardzugriffsprotokoll für Verzeichnisdatenbanken (z. B. Active Directory von Microsoft oder das quell-offene OpenLDAP),
- *NetInfo*, das ursprünglich von Next entwickelte Verzeichnisdatenbanksystem,
- *NIS*, ein auf älteren Unix-basierten Systemen verbreitetes Verzeichnisdatenbanksystem von Sun,

Abb. 4.1. Welche Verzeichnisdienste nun tatsächlich verwendet werden sollen, kann der Anwender mit Hilfe des Programms *Verzeichnisdienste* festlegen.

- *Lokale Konfigurationsdateien*, kein „Verzeichnisdienst“ im eigentlichen Sinne, sondern die Option, traditionelle Konfigurationsdateien als Open-Directory-Datenquellen zu verwenden.

Welche Verzeichnisdienste nun tatsächlich verwendet werden sollen, kann der Anwender mit Hilfe des Dienstprogramms *Verzeichnisdienste* festlegen. Im Bereich *Dienste* können der Verzeichnisdienst Active Directory, der Pseudo-Verzeichnisdienst „BSD-Konfigurationsdatei und NIS“, der Verzeichnisdienst LDAP und der Verzeichnisdienst NetInfo wahlweise aktiviert und konfiguriert werden (Abb. 4.1).

Im Bereich *Identifizierung* kann die Authentifizierung gegen einen oder mehrere Verzeichnisdienste aktiviert werden. Im Bereich Kontakte kann man definieren, welche der konfigurierten Verzeichnisdienste für die Suche nach Kontaktinformationen in Programmen wie Apple Mail oder Addressbuch verwendet werden sollen.

Die übrigen Einträge in der Diensteliste – AppleTalk, Bonjour, SLP und SMB/CIFS – können weder für Authentifizierung noch für die Suche nach Kontakten verwendet werden. Bei diesen Diensten handelt es sich vielmehr um Verfahren, mit denen Netzwerkanwendungen, insbesondere z. B. Fileserver, ihre Dienste im Netzwerken ankündigen können.

Der Verzeichnisdienst NetInfo hat in Mac OS X eine Sonderstellung. Die lokale NetInfo-Datenbank wird im Auslieferungszustand von Mac OS X unabhängig von den mit Hilfe des Programms Verzeichnisdienste vorgenommenen Einstellungen immer verwendet. Insbesondere sucht Mac OS X immer zuerst dort nach Benutzer- und Gruppeneinträgen.

Im Programm Verzeichnisdienste lässt sich lediglich beeinflussen, ob Mac OS X *zusätzlich* eine (zentrale) NetInfo-Verzeichnisdatenbank auf einem anderen Rechner konsultieren soll. Eine Abschaltung der lokalen NetInfo-Datenbank ist auf diese Weise nicht möglich.

4.1 NetInfo

Ein NetInfo-Verzeichnisdienst besteht grundsätzlich aus mindestens zwei Komponenten: der Verzeichnisdatenbank selbst und dem dazugehörigen Serverprogramm (realisiert mit Hilfe von `netinfod`), welches Anfragen von NetInfo-Clients (heutzutage in der Regel Open Directory) bearbeiten kann.

Unter Mac OS X werden NetInfo-Verzeichnisdatenbanken normalerweise im Verzeichnis `/var/db/netinfo/` abgelegt. Jeder Datenbank entspricht dabei ein Unterverzeichnis mit der Namensendung `.nidb` – so werden z. B. alle Daten der lokalen NetInfo-Datenbank im Unterverzeichnis `local.nidb` abgelegt.

Der Inhalt der Datenbank ist nur für root lesbar. Normalerweise wird die Datenbank ohnehin nicht direkt über das Dateisystem manipuliert, sondern mittelbar über die der Datenbank eindeutig zugeordnete Instanz von

`netinfod`. Eine wesentliche Aufgabe von `netinfod` ist dabei die Synchronisation von parallelen Zugriffen von NetInfo-Clients und damit die Sicherstellung der Datenintegrität.

Grundsätzlich kann auf eine NetInfo-Datenbank bzw. auf die entsprechende Instanz von `netinfod` sowohl lokal, als auch über das Netzwerk zugegriffen werden. Entsprechende Zugriffsrechte vorausgesetzt, lassen sich die Inhalte einer NetInfo-Datenbank also auch über das Netzwerk lesen und schreiben. Im Auslieferungszustand von Mac OS X 10.4 ist der Netzwerkzugriff auf die lokale NetInfo-Datenbank nicht möglich. Der Lesezugriff über das Netzwerk ist dabei natürlich die Voraussetzung für die Nutzung von NetInfo als Verzeichnisdienst.

Ein Mac-OS-X-Rechner kann ohne weiteres mehrere NetInfo-Datenbanken (mit entsprechend mehreren Instanzen von `netinfod`) beherbergen. Um mehrere, auf einem Rechner abgelegte Datenbanken unterscheiden zu können, wird jeder Datenbank ein eindeutiges Kennzeichen, engl. *Tag*, zugeordnet. Beim Anlegen der Datenbank muss dieses Kennzeichen angegeben werden und legt u. a. den Namen fest, unter dem die Datenbankdateien im Dateisystem abgelegt werden. Entsprechend lautet das Kennzeichen bzw. der *Tag* der lokalen NetInfo-Datenbank schlicht `local`.

NetInfo-Datenbanken lassen sich zu Datenbank-Hierarchien verketten. Im einfachsten (aber durchaus typischen) Fall besteht eine solche Hierarchie aus einer einzelnen zentralen Datenbank, welche die Wurzel der Hierarchie bildet, und einer größeren Anzahl von lokalen Datenbanken, die an die zentrale Datenbank *angebunden* wurden.

Jede NetInfo-Datenbank, die in eine solche Datenbank-Hierarchie eingebunden wurde, ist für einen Knoten der Hierarchie zuständig, wobei die Knoten als Domänen, engl. *Domains*, bezeichnet werden. Der Wurzelknoten wird entsprechend als Netzwerk-, Root- oder /-Domäne bezeichnet. Die lokalen Domänen der angeschlossenen Rechner werden mit dem jeweiligen Hostnamen bezeichnet.

Die Domänenbezeichnung ist grundsätzlich von dem Kennzeichen bzw. Tag der Datenbank unabhängig, wobei für die /-Domäne das Kennzeichen `network` und für die Blätter der Datenbankhierarchie, also die lokalen Domänen der an die Hierarchie angeschlossenen Rechner, das Kennzeichen `local` vorgesehen ist.

4.1.1 Datenbankstruktur

Jede NetInfo-Datenbank besteht aus einer Hierarchie von Einträgen. Jeder Datenbankeintrag bzw. Knoten wird durch ein Verzeichnis (engl. Directory) symbolisiert, welches optional mit Attributen (engl. Properties) versehen werden kann. Jedes Verzeichnis kann *zusätzlich* beliebig viele Unterverzeichnisse beinhalten, die ihrerseits wiederum mit Attributen versehen werden können.

NetInfo-Attribute haben eine Bezeichnung (engl. Property Key) und keinen, genau einen oder mehrere Werte (engl. Property Value). In der Regel

besitzt jedes Verzeichnis mindestens ein Attribut mit der Bezeichnung `name`, welches den Namen des jeweiligen Datenbankeintrags bzw. Verzeichnisses als Wert beinhaltet.

Eine typische NetInfo-Datenbank beinhaltet eine Reihe von Standard-Datenbankeinträgen. Diese Einträge werden nach Eintragtyp in eigenen Unterverzeichnissen zusammengefasst, so z. B. alle Benutzereinträge im Unterverzeichnis `users` (statt in der Datei `/etc/passwd`), alle Gruppeneinträge im Unterverzeichnis `groups` (statt in der Datei `/etc/groups`) und alle Hosteinträge im Unterverzeichnis `machines` (statt in der Datei `/etc/hosts`).

Ein Benutzereintrag selbst besteht dann entsprechend aus einem leeren Unterverzeichnis im NetInfo-Verzeichnis `/users/` und ist mit mehreren Attributen versehen: u. a. `name` (Login-ID bzw. Kurzname), `passwd` (Passwort), `uid` (Benutzernummer) und `home` (Benutzerverzeichnis).

4.1.2 Zugriffsrechte

NetInfo verfügt über eine in sich abgeschlossene, eigene Benutzerverwaltung. Zur Laufzeit lässt NetInfo nur authentifizierte Schreibzugriffe zu und verwendet zur Authentifizierung der Schreibzugriffe seine eigene, im Unterverzeichnis `/users` definierte Benutzerliste. Grundsätzlich ist ohne weitere Konfiguration ausschließlich der Benutzer *root* schreibberechtigt.

Um weiteren Benutzern den Schreibzugriff auf bestimmte Einträge oder bestimmte Attribute zu ermöglichen, besteht die Möglichkeit, über die Attribute `_writers` bzw. `_writers_attributname` eine Liste von schreibberechtigten Benutzern zu definieren.

Mit dem Attribut `_writers` definiert man dabei die Liste der schreibberechtigten Benutzer für ein komplettes Unterverzeichnis, während man mit dem Attribut `_writers_attributname` die Schreibberechtigung für ein einzelnes Attribut steuern kann. Letzteres wird z. B. verwendet, um einem Benutzer die Änderung seines eigenen Passworts zu erlauben. In diesem Fall heißt das Attribut dann entsprechend `_writers_passwd` und besitzt mit dem Login-Namen des schreibberechtigten Benutzers einen einzigen Wert.

Um mehreren Benutzern Schreibzugriff auf ein Unterverzeichnis oder ein Attribut zu gewähren, trägt man ihre Login-Namen einfach als unterschiedliche Werte der Attribute `_writers` bzw. `_writers_attributname` ein. Um allen (eingetragenen) Benutzern der NetInfo-Datenbank Schreibzugriff zu gewähren genügt die Angabe eines `*` als einzigen Wert.

Grundsätzlich ist der Lese- und (nach entsprechender Authentifizierung) Schreibzugriff auf die Inhalte der NetInfo-Datenbank sowohl lokal als auch über das Netzwerk möglich[1]. Der Netzwerkzugriff lässt sich allerdings zur Erhöhung der Sicherheit deaktivieren oder auf „vertrauenswürdige“ IP-Adressen beschränken. Man ergänzt dazu den Wurzeleintrag (`/`) der Datenbank um

[1] Unter Mac OS X 10.4 läuft NetInfo normalerweise in einer speziellen lokalen Betriebsart, die keine Netzwerkzugriffe unterstützt.

das Attribut `trusted_networks` und trägt als Werte die gewünschten Host- oder Netzwerkadressen ein. Falls das Attribut `trusted_networks` existiert, aber über keine Werte verfügt, wird der Netzwerkzugriff komplett deaktiviert.

4.1.3 Lokale Datenbank bearbeiten

Das Dienstprogramm *NetInfo Manager* erlaubt den direkten Zugriff auf lokale und auf entfernte NetInfo-Datenbanken (Abb. 4.2). Das Programm verwendet eine Spaltenansicht, um die hierarchische Struktur der NetInfo-Datenbank zu visualisieren. Die Attribute eines in der Spaltenansicht ausgewählten Elements werden im unteren Teil des Programmfensters angezeigt.

Der Benutzer kann mit Hilfe des *NetInfo Manager* neue Verzeichnisse (Menüeintrag *Neuer Unterordner* im Menü Ordner), neue Attribute (Menüeintrag *Neue Eigenschaft*) und neue Werte (Menüeintrag *Neuer Wert*) anle-

Abb. 4.2. Das Dienstprogramm *NetInfo Manager* erlaubt den direkten Zugriff auf lokale und auf entfernte NetInfo-Datenbanken.

gen sowie Verzeichnisse, Attribute und Werte löschen. Der Inhalt beliebiger Werte kann ebenfalls geändert werden. Natürlich ist dazu eine Authentifizierung als Administrator erforderlich.

Selbstverständlich lassen sich NetInfo-Datenbanken auch von der Kommandozeile aus manipulieren. Unter Mac OS X stehen hierzu die Befehle `niutil` und `nicl` zur Verfügung. Mit `niutil` (NetInfo Utility/Dienstprogramm) lassen sich beliebige Daten – eine erfolgreiche Authentifizierung vorausgesetzt – aus einer NetInfo-Datenbank herauslesen bzw. in eine NetInfo-Datenbank hineinschreiben. Um den Inhalt des Verzeichnisses `/users` der lokalen NetInfo-Datenbank auszulesen, muss man diesen Befehl z. B. wie folgt verwenden:

```
$ niutil -list / /users
11      nobody
12      root
13      daemon
14      unknown
15      lp
16      postfix
17      www
18      eppc
19      mysql
20      sshd
21      qtss
22      cyrusimap
23      mailman
24      appserver
25      clamav
26      amavisd
27      jabber
28      xgridcontroller
29      xgridagent
30      appowner
31      windowserver
32      tokend
33      securityagent
91      admin
```

Im Prinzip lässt sich das gleiche fast genauso auch mit dem Befehl `nicl` bewerkstelligen:

```
$ nicl / -list /users
11        nobody
12        root
13        daemon
14        unknown
15        lp
16        postfix
17        www
18        eppc
19        mysql
20        sshd
21        qtss
22        cyrusimap
23        mailman
24        appserver
25        clamav
```

```
26          amavisd
27          jabber
28          xgridcontroller
29          xgridagent
30          appowner
31          windowserver
32          tokend
33          securityagent
91          admin
```

Abgesehen von der leicht veränderten Reihenfolge der Parameter weist der Befehl `nicl` gegenüber `niutil` zwei Besonderheiten auf: der Befehl `nicl` verfügt über eine „interaktive“ Betriebsart, in der er als eine Art „Shell“ für die NetInfo-Datenbank fungieren kann, und er erlaubt den unmittelbaren Zugriff auf die Datenbank auf der Dateisystemebene.

Die interaktive Betriebsart wird aktiviert, indem man als Parameter einfach eine NetInfo-Datenbank ohne ein auszuführendes Kommando angibt. In dieser Betriebsart kann man ähnlich wie in einer Shell mit `ls` den Inhalt von Verzeichnissen ausgeben und mit `cd` zwischen Verzeichnissen wechseln. Mit `read` lassen sich schließlich die Attribute eines Verzeichnisses ausgeben:

```
$ nicl /
/ > ls
1           users
2           groups
3           machines
4           networks
5           protocols
6           rpcs
7           services
8           aliases
9           mounts
10          printers
92          config
98          afpuser_aliases
100         exports
/ > cd users
/users > ls
11          nobody
12          root
13          daemon
14          unknown
15          lp
16          postfix
17          www
18          eppc
19          mysql
20          sshd
21          qtss
22          cyrusimap
23          mailman
24          appserver
25          clamav
26          amavisd
27          jabber
28          xgridcontroller
29          xgridagent
```

```
30          appowner
31          windowserver
32          tokend
33          securityagent
91          admin
103         ftp
105         rkk
/users > read admin
passwd: ********
uid: 501
generateduid: D645CC33-3BD9-4A8A-8C92-66EDC0063786
home: /Users/admin
shell: /bin/bash
hint:
gid: 20
authentication_authority: ;ShadowHash;HASHLIST: <SALTED-SHA1,
                                        SMB-NT, SMB-LAN-MANAGER>
name: admin
picture: /Library/User Pictures/Flowers/Flower.tif
realname: Administrator
_writers_hint: admin
_writers_picture: admin
_writers_tim_password: admin
_shadow_passwd:
_writers_passwd: admin
_writers_realname: admin
/users >
```

Ein solcher Aufruf von `nicl` verwendet, wie alle anderen NetInfo-Clients, die zuständige Instanz von `netinfod`, um auf die NetInfo-Datenbank zuzugreifen. Alternativ sieht `nicl` aber auch den umittelbaren Zugriff unter Umgehung von `netinfod` vor und erlaubt direkte Zugriffe auf die auf dem lokalen Dateisystem gespeicherten NetInfo-Daten:

```
$ ls -l /var/db/netinfo/
total 0
drwx------   21 root  wheel  714 Jul 30 17:42 local.nidb
$ sudo nicl -raw /var/db/netinfo/local.nidb/
/ > ls
1           users
2           groups
3           machines
4           networks
5           protocols
6           rpcs
7           services
8           aliases
9           mounts
10          printers
92          config
98          afpuser_aliases
100         exports
```

Diese Art von Zugriff umgeht die normale NetInfo-Zugriffssteuerung und erfordert root-Rechte, da die NetInfo-Daten nur für root lesbar sind. Im laufenden Betrieb sollte diese Methode nicht verwendet werden, da in diesem Fall parallele Schreibzugriffe von nicl und anderen Programmen nicht synchronisiert werden und in einer inkonsistenten und damit unbrauchbaren NetInfo-

Datenbank resultieren können. Sofern die NetInfo-Infrastruktur noch nicht (oder nicht mehr) läuft, insbesondere im Single-User-Modus, stellt diese Methode jedoch die einzige Möglichkeit dar, tatsächlich Veränderungen in der NetInfo-Datenbank zu bewirken.

Neben den vielseitig verwendbaren Befehlen `niutil` und `nicl` stehen unter Mac OS X noch einige spezialisierte Befehle zur Datenabfrage (`nigrep`, `nireport` und `nifind`) sowie für den Austausch von Daten mit anderen Unix-basierten Systemen (`nidump` und `niload`) zur Verfügung.

Mit `nigrep` lassen sich die Attribute einer Datenbank bequem nach einem regulären Ausdruck durchsuchen. Es lassen sich damit also z. B. alle Attribute nach der Zeichenkette `root` durchsuchen:

```
$ nigrep root /
12 /users/root:  name root
12 /users/root:  home /var/root
12 /users/root:  _writers_passwd root
13 /users/daemon:  home /var/root
36 /groups/wheel:  users root
37 /groups/daemon:  users root
38 /groups/kmem:  users root
39 /groups/sys:  users root
40 /groups/tty:  users root
41 /groups/operator:  users root
44 /groups/staff:  users root admin
48 /groups/certusers:  users root jabber postfix cyrusimap
59 /groups/admin:  users root admin
75 /aliases/administrator:  members root
76 /aliases/postmaster:  members root
79 /aliases/nobody:  members root
80 /aliases/dumper:  members root
81 /aliases/manager:  members root
82 /aliases/operator:  members root
```

Mit `nireport` kann man Attribute eines NetInfo-Verzeichnisses gezielt in Tabellenform ausgeben lassen:

```
$ nireport / /users uid name shell
-2      nobody  /usr/bin/false
0       root    /bin/sh
1       daemon  /usr/bin/false
99      unknown /usr/bin/false
26      lp      /usr/bin/false
27      postfix /usr/bin/false
70      www     /usr/bin/false
71      eppc    /usr/bin/false
74      mysql   /usr/bin/false
75      sshd    /usr/bin/false
76      qtss    /usr/bin/false
77      cyrusimap       /usr/bin/false
78      mailman /usr/bin/false
79      appserver       /usr/bin/false
82      clamav  /bin/tcsh
83      amavisd /bin/tcsh
84      jabber  /usr/bin/false
85      xgridcontroller /usr/bin/false
86      xgridagent      /usr/bin/false
87      appowner        /usr/bin/false
```

```
88      windowserver    /usr/bin/false
91      tokend  /usr/bin/false
92      securityagent   /usr/bin/false
501     admin   /bin/bash
```

Mit `nifind` kann man schließlich gezielt nach dem Vorkommen eines Verzeichnisses in einer NetInfo-Verzeichnishierarchie suchen:

```
$ nifind /users/admin  /
/users/admin found in /, id = 91
```

Um Konfigurationsdaten etwas bequemer mit anderen Unix-basierten Systemen austauschen zu können, stehen die Befehle `nidump` und `niload` zur Verfügung. Mitttels `nidump` lassen sich die in der NetInfo-Datenbank gespeicherten Informationen in dem jeweils auf Unix-basierten Systemen gebräuchlichen Textformat ausgeben, insbesondere beispielsweise Benutzerdaten im Format der Datei `/etc/passwd` und Gruppendaten im Format der Datei `/etc/group`[2]:

```
$ nidump passwd /
nobody:*:-2:-2::0:0:Unprivileged User:/var/empty:/usr/bin/false
root:********:0:0::0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1::0:0:System Services:/var/root:/usr/bin/false
unknown:*:99:99::0:0:Unknown User:/var/empty:/usr/bin/false
lp:*:26:26::0:0:Printing Services:/var/spool/cups:/usr/bin/false
postfix:*:27:27::0:0:Postfix User:/var/spool/postfix:/usr/bin/false
www:*:70:70::0:0:World Wide Web Server:/Library/WebServer:
                                        /usr/bin/false
eppc:*:71:71::0:0:Apple Events User:/var/empty:/usr/bin/false
mysql:*:74:74::0:0:MySQL Server:/var/empty:/usr/bin/false
sshd:*:75:75::0:0:sshd Privilege separation:/var/empty:
                                           /usr/bin/false
qtss:*:76:76::0:0:QuickTime Streaming Server:/var/empty:
                                             /usr/bin/false
cyrusimap:*:77:6::0:0:Cyrus IMAP User:/var/imap:/usr/bin/false
mailman:*:78:78::0:0:Mailman user:/var/empty:/usr/bin/false
appserver:*:79:79::0:0:Application Server:/var/empty:/usr/bin/false
clamav:*:82:82::0:0:Clamav User:/var/virusmails:/bin/tcsh
amavisd:*:83:83::0:0:Amavisd User:/var/virusmails:/bin/tcsh
jabber:*:84:84::0:0:Jabber User:/var/empty:/usr/bin/false
xgridcontroller:*:85:85::0:0:Xgrid Controller:/var/xgrid
                                               /controller:
                                               /usr/bin/false
xgridagent:*:86:86::0:0:Xgrid Agent:/var/xgrid/agent:/usr/bin/false
appowner:*:87:87::0:0:Application Owner:/var/empty:/usr/bin/false
windowserver:*:88:88::0:0:WindowServer:/var/empty:/usr/bin/false
tokend:*:91:91::0:0:Token Daemon:/var/empty:/usr/bin/false
securityagent:*:92:92::0:0:SecurityAgent:/var/empty:/usr/bin/false
admin:********:501:20::0:0:Administrator:/Users/admin:/bin/bash

$ nidump group /
nobody:*:-2:
```

[2] Mac OS X 10.3 verwendet im Gegensatz zu früheren Versionen im Normalfall keine Crypt-Hashes für Passwörter. Ein Export der Passwort-Hashes mit nidump ist daher nicht mehr möglich.

```
nogroup:*:-1:
wheel:*:0:root
daemon:*:1:root
kmem:*:2:root
sys:*:3:root
tty:*:4:root
operator:*:5:root
mail:*:6:
bin:*:7:
staff:*:20:root,admin
lp:*:26:
postfix:*:27:
postdrop:*:28:
certusers:*:29:root,jabber,postfix,cyrusimap
utmp:*:45:
uucp:*:66:
dialer:*:68:
network:*:69:
www:*:70:
mysql:*:74:
sshd:*:75:
qtss:*:76:
mailman:*:78:
appserverusr:*:79:admin
admin:*:80:root,admin
appserveradm:*:81:admin
clamav:*:82:
amavisd:*:83:
jabber:*:84:
xgridcontroller:*:85:
xgridagent:*:86:
appowner:*:87:
windowserver:*:88:
accessibility:*:90:
tokend:*:91:
securityagent:*:92:
unknown:*:99:
everyone::12:
authedusers::50:
interactusers::51:
netusers::52:
consoleusers::53:
owner::10:
group::16:
smmsp::25:
```

4.1.4 Netzwerk-Datenbank erstellen

Obwohl NetInfo-Netzwerk-Datenbanken seit der Version 10.3 nicht mehr offiziell unterstützt werden (u. a. wurden die entsprechenden Menübefehle aus dem *NetInfo Manager* entfernt), lassen sich solche Datenbanken noch immer manuell, mit Hilfe der Kommandozeile, problemlos einrichten.

Im Auslieferungszustand von Mac OS X 10.3 und 10.4 läuft das NetInfo-System in einer besonderen lokalen Betriebsart, die grundsätzlich keinen Netzwerk-Zugriff auf lokal betriebene NetInfo-Datenbanken erlaubt:

```
$ nidomain -l
The "nibindd" daemon is not running on this desktop system.
Port information is not available.
```

Um einen solchen Zugriff dennoch zu ermöglichen, muss das NetInfo-System in der in früheren Versionen von Mac OS X üblichen Netzwerk-Betriebsart gestartet werden.

In Mac OS X 10.3 fragt das Skript `/etc/rc` die Variable `NETINFOSERVER` ab und startet abhängig von deren Wert die für die Netzwerk-Betriebsart notwendigen Prozesse. Der Wert dieser Variablen kann in der Datei `/etc/hostconfig` vom Administrator gesetzt werden. Um eine neue, zusätzliche NetInfo-Datenbank anlegen zu können, muss man den Wert dieser Variablen daher auf den Wert `-YES-` setzen und einen Neustart durchführen.

In Mac OS X 10.4 wird die Wahl zwischen der lokalen Betriebsart und der Netzwerk-Betriebsart innerhalb von `nibindd` getroffen. Normalerweise startet `nibindd` immer in der lokalen Betriebsart. Im öffentlich zugänglichen Quellcode (Abb. 4.3) von `nibindd` wird die Option `-A` abgefragt, die dem Anschein nach die Netzwerk-Betriebsart wieder aktivieren bzw. die Aktivierung der lokalen Betriebsart verhindern kann (Datei nibind_main.c – Stand Darwin 8.2 bzw. Mac OS X 10.4.2):

```
...

int
main(int argc, char *argv[])
{

...

localonly = 1;

...
if (!strcmp(argv[i], "-A"))
{
localonly = 0;
}
...

if (localonly)
{
#ifdef _NETINFO_FLOCK_
closedir(dp);
#endif
system_log_close();
execl(NETINFO_PROG, NETINFO_PROG, "-s", "local", NULL);
system_log(LOG_ALERT, "execl(\"%s\"): %m", NETINFO_PROG);
exit(1);
}

...
```

Um `nibindd` mit dieser Option zu starten, muss man unter Mac OS X 10.4 die entsprechende `launchd`-Konfigurationsdatei ändern und im Abschnitt

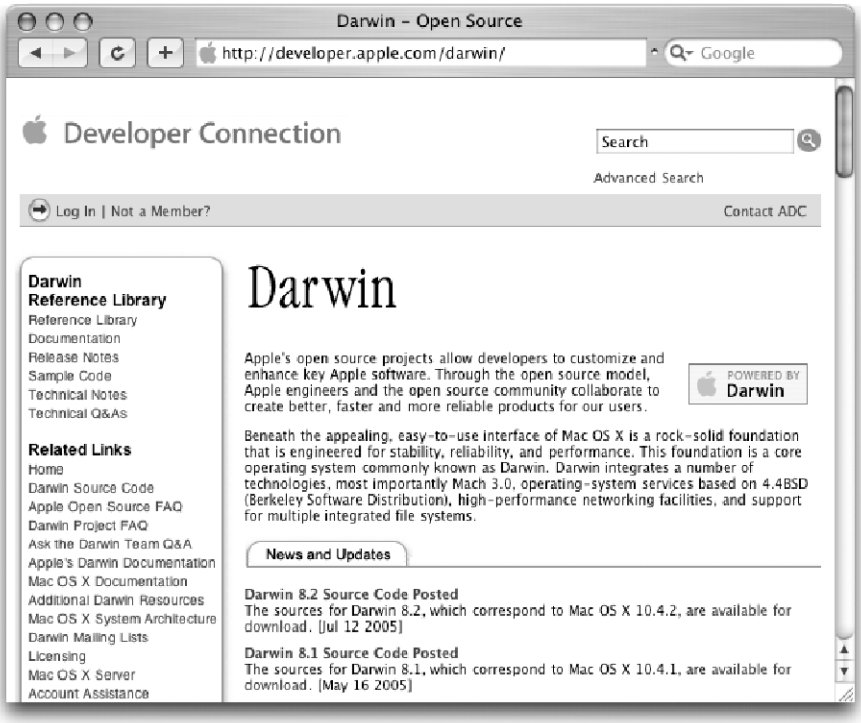


Abb. 4.3. Der Quelltext des Betriebssystemkerns und der Unix-Benutzerumgebung ist öffentlich zugänglich.

`ProgramArguments` den Wert `-A` einfügen. Danach muss man das System neu starten:

```
$ cat /System/Library/LaunchDaemons/com.apple.nibindd.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
            "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.apple.nibindd</string>
    <key>OnDemand</key>
    <false/>
    <key>ServiceIPC</key>
    <false/>
    <key>ProgramArguments</key>
    <array>
            <string>/usr/sbin/nibindd</string>
            <string>-A</string>
    </array>
</dict>
</plist>
```

Nach der Änderung der NetInfo-Konfiguration kann man mit Hilfe des `nidomain`-Befehls eine neue NetInfo-Datenbank anlegen. Die Option `-l` listet alle auf einem bestimmten Host gespeicherten NetInfo-Datenbanken auf:

```
$ nidomain -l
tag=local udp=1033 tcp=1033
```

Die Option `-m` erzeugt hingegen eine neue, leere NetInfo-Datenbank. Als Parameter muss ein eindeutiges Kennzeichen bzw. der *Tag* der Datenbank angegeben werden. Das Kennzeichen der hierarchisch am höchsten angesiedelten Wurzel-Datenbank lautet dabei normalerweise `network`:

```
$ sudo nidomain -m network

$ nidomain -l
tag=local udp=1033 tcp=1033
tag=network udp=1000 tcp=810

$ ls -l /var/db/netinfo/
total 0
drwx------  13 root  wheel  442 16 Jan 14:12 local.nidb
drwx------   8 root  wheel  272 16 Jan 14:28 network.nidb
```

Um auf diese neue NetInfo-Datenbank mit NetInfo-Werkzeugen zugreifen zu können, muss man anschließend nur noch die Neuinitialisierung des NetInfo-Datenbanksystems veranlassen, indem man ein SIGHUP-Signal an `nibindd` sendet oder einen Neustart durchführt. Danach kann man z. B. mit nicl oder mit dem NetInfo-Manager unter Angabe der Hostadresse und des Kennzeichens auf den Inhalt der Datenbank von einem beliebigen Rechner aus zugreifen:

```
$ nicl -t localhost/network -list /
1          machines
```

Die neue Datenbank ist bis auf einen einzigen Eintrag im Verzeichnis `/machines` leer:

```
 $ nicl -t localhost/network -list /machines/
2          Tiger-Server.local

$ nicl -t localhost/network -read /machines/
                                  Tiger-Server.local
name: Tiger-Server.local
ip_address: 192.168.0.10 192.168.2.1
serves: ./network
```

Dieser Eintrag weist die an diese NetInfo-Datenbank eventuell später gebundenen Clients auf den Namen sowie die IP-Adresse des Hosts hin, der die Originalversion der Datenbank beherbergt.

Um weitere Einträge zu erzeugen, muss man sich gegenüber dem NetInfo-Datenbanksystem mit dem Namen und dem Passwort eines schreibberechtigten Benutzers ausweisen. Das ist normalerweise der Benutzer `/users/root`, der in der neuangelegten Datenbank jedoch fehlt.

Damit sind jegliche Schreibzugriffe auf die neuangelegte NetInfo-Datenbank zunächst mal ausgeschlossen. Eine Ausnahme bilden lediglich direkte, lokale Schreibzugriffe auf die Datenbankdateien mit Hilfe der `-raw`-Option von `nicl` oder mit `niutil`. Mit Hilfe dieser Befehle lässt sich insbesondere ein Eintrag für den Benutzer `root` anlegen, mit dessen Hilfe Schreibzugriffe dann später auch mit „normalen“ Mitteln (z. B. mit dem NetInfo-Manager) erledigt werden können.

Um den Benutzer `root` anzulegen, legt man zunächst auf der obersten Ebene der NetInfo-Datenbank ein Unterverzeichnis `/users` an und legt darin ein weiteres Unterverzeichnis für den eigentlichen Benutzereintrag an. Der Benutzereintrag muss mindestens über die Attribute `uid` (Benutzer-ID), `guid` (Gruppen-ID) und `passwd` (Passwort, anfangs leer[3]) verfügen:

```
$ sudo  niutil -create -t localhost/network /users

$ nicl -t localhost/network -list /
1          machines
3          users

$ sudo  niutil -create -t localhost/network /users/root

$ nicl -t localhost/network -list /users
4          root

$ sudo niutil -createprop -t localhost/network /users/
                                                root uid 0
$ sudo niutil -createprop -t localhost/network /users/
                                                root gid 0
$ sudo niutil -createprop -t localhost/network
                              /users/root passwd ""

$ nicl -t localhost/network -read /users/root
name: root
uid: 0
gid: 0
passwd:
```

Natürlich sollte das Passwort des Benutzers `root`, der uneingeschränkte Schreibrechte auf der NetInfo-Datenbank besitzt, am besten gleich geändert werden. Am einfachsten lässt sich das mit Hilfe des Kommandos `passwd` bewerkstelligen (im Beispiel wurde `root1` als Password verwendet – verifizierbar mit `openssl passwd -salt rX root1`):

```
$ sudo passwd -i netinfo -l imac/network root
Changing password for root.
New password:
Retype new password:

$ nicl -t localhost/network -read /users/root
name: root
uid: 0
gid: 0
passwd: rXykHONXvgPrU
```

[3] Hier könnte man statt eines leeren Passworts auch einen Crypt-Hash einfügen, den man zuvor z. B. mit Hilfe von openssl passwd erstellt oder aus einer vorhandenen passwd-Datei kopiert hat.

Abb. 4.4. Mit dem Befehl *Nach Kennung öffnen...* (Menü *Domain*) kann man auf im Netzwerk freigegebene Datenbanken zugreifen.

Alternativ lässt sich das gleiche Ergebnis natürlich durch die manuelle Erstellung eines eigenen Crypt-Hashes erzielen:

```
$ openssl passwd root1
Eqikq.fBlnp/M

$ sudo niutil -createprop -t imac/network /users/root
                          passwd "Eqikq.fBlnp/M"
$ nicl -t localhost/network -read /users/rootname: root
uid: 0
gid: 0
passwd: Eqikq.fBlnp/M
```

Nachdem man einen Eintrag für den Benutzer `root` erstellt hat, kann man bequem mit Hilfe des NetInfo-Managers auf die neue Datenbank zugreifen. Man ruft das Programm dazu auf einem beliebigen Rechner auf und wählt im Menü *Domain* den Befehl *Nach Kennung öffnen ...*[4].

Im daraufhin erscheinenden Dialogfenster sind dann der Rechnername[5] oder die IP-Adresse des die Datenbank beherbergenden Rechners sowie das Kennzeichen (bzw. *Tag*) der Datenbank einzutragen (siehe Abb. 4.4).

Hat man den korrekten Hostnamen und ein passendes Kennzeichen eingegeben, erscheint der gewohnte Browser, mit dem man bequem den Inhalt der Datenbank inspizieren und gegebenenfalls auch Veränderungen durchführen kann (Abb. 4.5). Um Veränderungen durchführen zu können, muss man sich zuvor über einen Klick auf das Schloss-Symbol in der linken unteren Ecke des NetInfo-Browsers natürlich wie üblich authentifizieren. Zur Authentifizierung muss als Benutzername `root` und als Passwort das zuvor definierte Passwort eingegeben werden (im obigen Beispiel `root1`).

4.1.5 Netzwerkbenutzer anlegen

Mit einer (fast) leeren NetInfo-Datenbank kann man nicht allzu viel anfangen. Die gängigste Anwendung einer im Netzwerk lesbaren Verzeichnisdatenbank

[4] Die entfernte Anmeldung kann sich verzögern, falls die IP-Adressen der beteiligten Rechner nicht korrekt aufgelöst werden können.

[5] Sofern ein DNS-Server verfügbar ist. Bonjour-Namen können an dieser Stelle z. Z. nicht verwendet werden.

Abb. 4.5. Der NetInfo Manager bietet eine einfache graphische Benutzeroberfläche mit der die neu angelegte NetInfo-Datenbank betrachtet werden kann.

ist die Bereitstellung eines Benutzerverzeichnisses, das als zentraler Katalog zur Identifizierung und zur Authentifizierung von Benutzern verwendet werden kann.

Um einen Netzwerkbenutzer anzulegen, kann man entweder auf die Befehlszeilenkommandos `niutil` bzw. `nicl` oder aber auf den NetInfo-Manager zurückgreifen. In jedem Fall muss für jeden Netzwerkbenutzer im Unterverzeichnis `/users` ein eigener Eintrag angelegt werden.

Mac OS X liest neben den bereits innerhalb des Eintrags für den Benutzer `root` verwendeten Attributen eine Reihe weiterer Attribute, wie man leicht anhand der Einträge in der lokalen NetInfo-Datenbank überprüfen kann. Ein Netzwerkbenutzer benötigt aber nicht unbedingt alle Attribute, die für lokale Benutzer verwendet werden. Tabelle 4.1 fasst die wichtigsten Attribute zusammen, die man beim Anlegen eines Netzwerkbenutzers verwenden sollte.

Die in der NetInfo-Datenbank gespeicherten Attribute können grundsätzlich nur vom Benutzer `root` geändert werden. Um dem Benutzer selbst die Änderung seines Passworts, seines Bilds sowie seines „langen“ Namens über den Benutzerbereich der Systemeinstellungen zu ermöglichen, muss man dem Benutzer deshalb explizit Schreibzugriff auf bestimmte Attribute gewähren. Um z. B. dem Benutzer `test` die Änderung des Passworts zu erlauben, muss

Tabelle 4.1. Die wichtigsten Attribute für Netzwerkbenutzer.

NetInfo-Attribut	Beschreibung
name	Eindeutiger Kurzname, z. B. `test`
uid	Eindeutige Benutzer-ID, z. B. 2048. Mac OS X verwendet für normale Benutzer fortlaufende IDs ab 501, Mac-OS-X-Server ab 1024.
gid	Gruppen-ID, z. B. 20 für die Standardgruppe `staff`
passwd	Passwort, z. B. `Lt15koe2s.g7Y` (Crypt-Hash für „test“)
hint	Merkhilfe für das Passwort
home	Privates Verzeichnis, z. B. `/Users/test`. Sofern dieses Verzeichnis noch nicht existiert, wird es bei der ersten Anmeldung angelegt.
shell	Bevorzugte Shell, normalerweise `/bin/bash`
picture	Bild des Benutzers, z. B. `/Library/User Pictures/Animals/Cat.tif`
realname	„Langer“ Name des Benutzers, z. B. `Test Benutzer`
SharedDir	freizugebendes Verzeichnis für Personal Filesharing, normalerweise `Public`

man demnach im Benutzereintrag das zusätzliche Attribut `_writers_passwd` mit dem Wert `test` erstellen.

Ein für einfache Anwendungen ausreichend vollständiger Eintrag für den Netzwerkbenutzer `test` könnte dann wie folgt aussehen:

```
$ nicl -t imac/network -read /users/test
passwd: Lt15koe2s.g7Y
uid: 2048
home: /Users/test
shell: /bin/bash
hint:
gid: 20
sharedDir: Public
picture: /Library/User Pictures/Animals/Dog.tif
name: test
realname: Test Benutzer
_writers_hint: test
_writers_picture: test
_writers_realname: test
_writers_passwd: test
```

4.1.6 Netzwerkdatenbank für Authentifizierung verwenden

Um eine im Netzwerk zugängliche NetInfo-Datenbank für die Authentifizierung von Benutzern verwenden zu können, muss man auf der Client-Seite zwei Konfigurationsschritte vornehmen. Als erstes muss man dem Client-Rechner sagen, dass er eine Verbindung zu einer bestimmten NetInfo-Datenbank aufbauen soll. Im zweiten Schritt muss man definieren, dass dort gefundene Benutzerdaten für die Authentifizierung von Benutzern verwendet werden

dürfen. Der zweite Konfigurationsschritt kann entfallen, wenn für die Authentifizierung der normalerweise voreingestellte „automatische“ Suchpfad aktiv ist.

Beide Konfigurationsschritte lassen sich mit dem Dienstprogramm *Verzeichnisdienste* (engl. *Directory Access*) bewerkstelligen.

Im ersten Konfigurationsschritt muss man im Bereich Dienste das Open Directory Plug-In für NetInfo aktivieren. Dazu muss man in der Diensteliste des Dienstprogramms Verzeichnisdienste die entsprechende Checkbox aktivieren, und das Plug-In mit den passenden Verbindungsparametern versehen (siehe Abb. 4.6).

Den Konfigurationsdialog des Plug-Ins erreicht man, indem man nach der Auswahl des Plug-Ins aus der Diensteliste auf die Schaltfläche *Konfigurieren* klickt (u. U. ist dafür eine Authentifizierung als Administrator erforderlich).

Im Konfigurationsdialog kann man die Art und Weise auswählen, auf die der Rechner die zu verwendende NetInfo-Datenbank finden soll (siehe

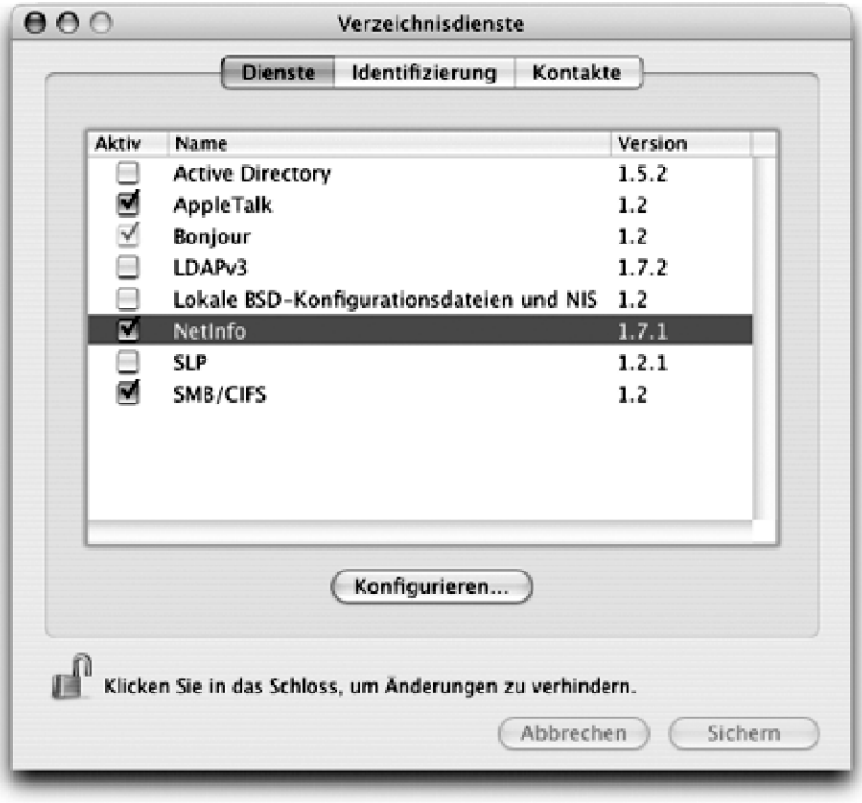


Abb. 4.6. Das Open Directory NetInfo-Plug-in wird im Dienstprogramm *Verzeichnisdienste* aktiviert und konfiguriert.

Abb. 4.7). Es stehen insgesamt drei Varianten zur Verfügung: die Broadcast-Methode, die DHCP-Methode und die direkte Methode.

Wird die Broadcast-Methode ausgewählt, sendet der Client beim Start eine Broadcast-Anfrage in das lokale Subnetz. Diese Anfrage wird automatisch von NetInfo-Servern beantwortet. Bedingung ist, dass sie über eine NetInfo-Datenbank verfügen, die mit dem Kennzeichen `network` versehen ist, und die einen zum Client passenden Eintrag im Verzeichnis `/machines` enthält.

Bei der DHCP-Methode wird die IP-Adresse des NetInfo-Servers und das Kennzeichen der zu verwendenden NetInfo-Datenbank mit Hilfe eines DHCP-Antwortpakets übermittelt. Damit diese Methode funktioniert, muss der verwendete DHCP-Server über eine entsprechende Konfigurationsmöglichkeit verfügen.

Die direkte Konfigurationsmethode erlaubt schließlich die direkte Angabe der IP-Adresse des NetInfo-Servers sowie des Kennzeichens einer von diesem Server bereitgestellten NetInfo-Datenbank.

Abb. 4.7. Im Konfigurationsdialog des NetInfo-Plug-Ins lässt sich die zu verwendende NetInfo-Datenbank definieren.

Nach der Durchführung aller erforderlichen Änderungen sollte man die Änderungen speichern und den aktuellen Benutzer abmelden, um zum Anmeldefenster zu gelangen.

Sofern das Anmeldefenster Textfelder für den Namen und das Passwort des Benutzers anzeigt, können die entsprechenden Angaben für die Netzwerkbenutzer direkt eingegeben werden. Zeigt das Anmeldefenster hingegen eine Liste der (lokalen) Benutzer, muss zuvor das Icon „Andere…“ aus dieser Liste ausgewählt werden.

Nach erfolgreicher Anmeldung wird für den Netzwerkbenutzer ein lokales Benutzerverzeichnis angelegt.

4.1.7 Zentrale Benutzerverzeichnisse anlegen

Die im letzten Abschnitt realisierte zentrale NetInfo-Datenbank erlaubt zwar die zentrale Speicherung der Benutzerattribute wie z. B. deren Passwörter, die Speicherung der Benutzerdaten erfolgt jedoch nach wie vor dezentral, auf den Client-Rechnern.

Um die Benutzerdaten ebenfalls zentral speichern zu können, muss man die privaten Verzeichnisse der Benutzer auf eine für alle zu verwendenden Client-Rechner zugängliche Freigabe (NFS oder AFP) kopieren sowie dafür sorgen, dass diese Freigabe bei jeder Anmeldung eines Netzwerkbenutzers mit den richtigen Zugriffsrechten aktiviert wird.

Die Freigaben mit den privaten Verzeichnissen lassen sich unter Mac OS X automatisch bei jedem Systemstart aktivieren, wenn man in der zentralen NetInfo-Datenbank entsprechende Einträge vom Typ `mounts` erstellt hat.

Diese so genannten *automount*-Einträge müssen vier Attribute enthalten: `name`, `dir`, `opts` und `vfstype`. Das Attribut `name` identifiziert die automatisch zu aktivierende Freigabe. Sein Wert setzt sich aus dem Servernamen und dem vollständigen Pfad zum auf dem Fileserver freigegebenen Ordner zusammen, z. B. `Tiger-Server.local:/Shares/Users`.

Das Attribut `dir` bezeichnet den Ort, an dem das freigegebene Verzeichnis auf der Client-Seite aktiviert werden soll. Für private Benutzerverzeichnisse verwendet man auf der Client-Seite hierzu normalerweise das Verzeichnis `/Network/Servers` (es handelt sich hierbei um so genannte *dynamische* Automounts). Andere gebräuchliche Werte für dieses Attribut lauten `/Network/Library` oder `/Network/Applications` und werden für *statische* Automounts verwendet.

Das Attribut `vfstype` definiert den Typ des für die Aktivierung der Freigabe zu verwendenden virtuellen Dateisystems. Praktisch definiert man mit diesem Attribut, ob es sich bei der Freigabe um ein NFS- (Wert `nfs`) oder um ein AFP-Dateisystem (Wert `url`) handelt.

Das Attribut `opts` definiert schließlich die Aktivierungsparameter. Für statisch aktivierte NFS-Freigaben wird dieses Attribut nicht benötigt. Der Wert `net` deklariert den Eintrag als dynamischen Automount, der Wert `url`

definiert den Pfad zur Freigabe unter Angabe des Servers und der Bezeichnung der Freigabe.

Ein typischer Eintrag für einen dynamischen Automount lautet demnach wie folgt (Abb. 4.8):

```
$ nidump -r /mounts/server.example.com:\\/Shares\\/Users /
{
  "name" = ( "server.example.com:/Shares/Users" );
  "dir" = ( "/Network/Servers" );
  "vfstype" = ( "url" );
  "opts" = ( "url==afp://;AUTH=NO%20USER%20AUTHENT@Tiger-
            Server.local/Users", "net" );
}
```

Ein solcher Eintrag lässt sich am einfachsten mit Hilfe des NetInfo-Managers erstellen. Voraussetzung ist natürlich, dass auf dem Rechner `Tiger-Server.local` tatsächlich ein Verzeichnis `/Shares/Users/` existiert.

Nachdem man die Verzeichnisse erstellt hat, kann man den Befehl `createhomedir` verwenden, um ein vorkonfiguriertes Standardbenutzerverzeichnis zu erstellen:

```
$ mkdir -p /Shares/Users

$ sudo createhomedir -u test
creating home directories for (Tiger-Server.local)
created (/Network/Servers/Tiger-Server.local/Shares/
         Users/test)
```

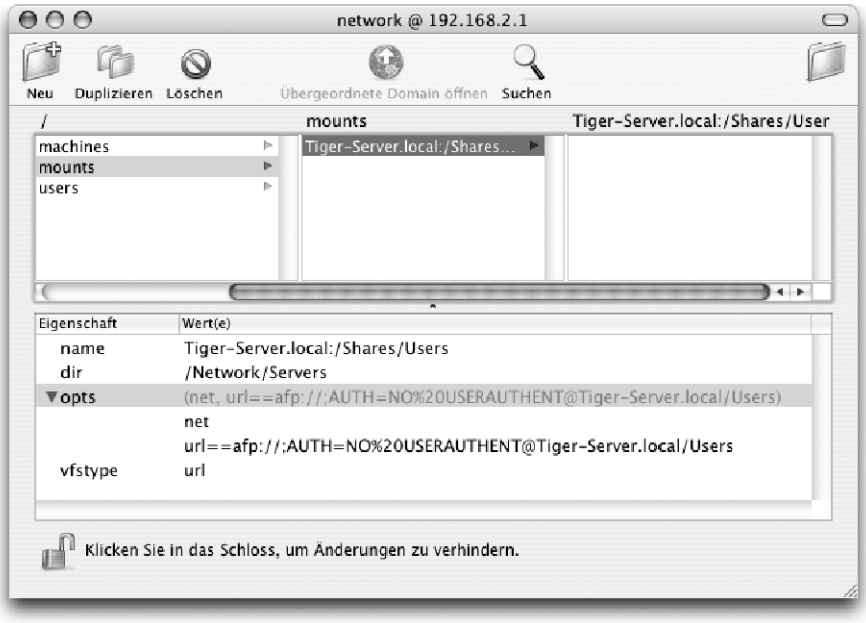


Abb. 4.8. Ein dynamischer Automount für Benutzerverzeichnisse.

Das funktioniert u. a. deshalb so einfach auf dem als Server verwendeten Rechner, weil auf jedem Mac OS X der folgende symbolische Link existiert:

```
$ ls -l /Network/Servers/
total 1
lrwxr-xr-x   1 root  wheel  512 Jul 30 20:40 Tiger-
             Server.local -> /
```

Damit die Netzwerkbenutzer auch auf dem Server funktionieren (und damit der AFP-Server sie überhaupt akzeptiert) muss man auf dem Serverrechner das Dienstprogramm Verzeichnisdienste starten und wie bereits weiter oben beschrieben NetInfo aktivieren und die network-Datenbank binden.

Anschließend muss man das Verzeichnis, welches die Benutzerverzeichnisse enthält, per AFP freigeben. Um eine solche Freigabe unter Mac OS X 10.4 einzurichten, muss man den folgenden Eintrag im NetInfo-Verzeichnis `/config/SharePoints` erstellen (Abb. 4.9):

```
$ nidump -r /config/SharePoints -t localhost/local
{
  "name" = ( "SharePoints" );
  CHILDREN = (
    {
      "afp_shared" = ( "1" );
      "name" = ( "Users" );
      "directory_path" = ( "/Shares/Users" );
    }
  )
}
```

Ein Client, der an eine NetInfo-Datenbank gebunden ist, die einen entsprechenden Automount-Eintrag enthält (oder dessen lokale NetInfo-Datenbank um einen solchen Eintrag ergänzt wurde), kann dann eine solche Freigabe automatisch aktivieren.

Da es sich in unserem Beispiel um einen dynamischen Automount handelt, erfolgt die Aktivierung dabei nicht sofort nach dem Start, sondern erst beim ersten Zugriff auf das entsprechende lokale Verzeichnis (im Beispiel `/Network/Servers/Tiger-Server.local/`). Im Vergleich zu statischen Automounts lassen sich auf diese Weise Ressourcen auf der Server-Seite sparen.

Um ein innerhalb einer solchen Freigabe abgelegtes Benutzerverzeichnis verwenden zu können, müssen natürlich zusätzlich noch die entsprechenden Angaben im Benutzereintrag ergänzt werden. Das Attribut `home` definiert dabei den lokalen Pfad zum privaten Benutzerverzeichnis auf dem Client-Rechner, z. B. für den Benutzer `test` bisher `/Users/test/`.

Um ein automatisch aktiviertes Verzeichnis verwenden zu können, muss dieser Wert entsprechend dem benutzten lokalen Pfad geändert werden, also z. B.:

```
$ nicl / -read /users/test home
home: /Network/Servers/Tiger-Server.local/Shares/Users/test
```

Würde das Benutzerverzeichnis auf einer NFS-Freigabe liegen, wäre das alles. Liegt das Benutzerverzeichnis allerdings (wie im obigen Beispiel) auf

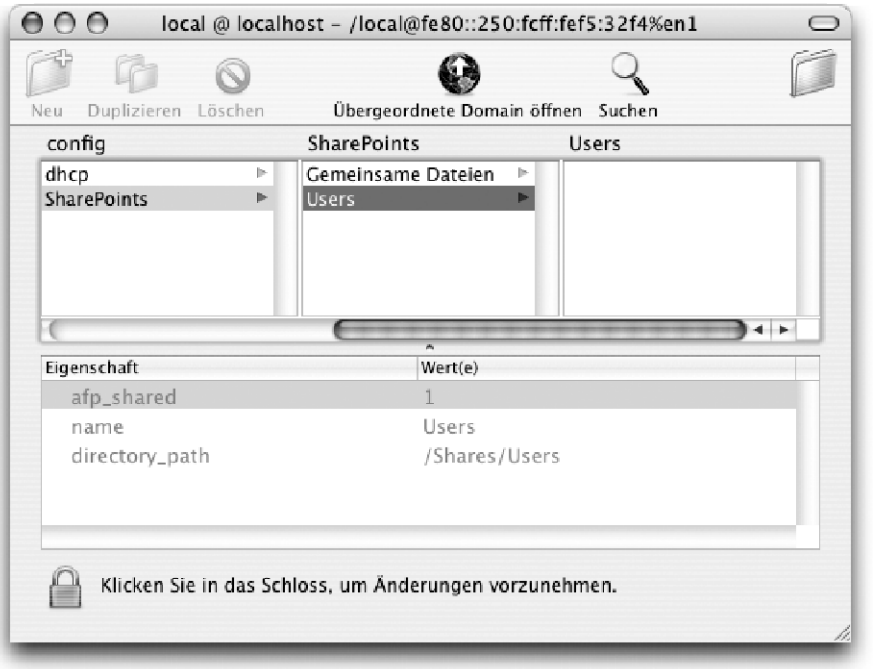


Abb. 4.9. Das die Benutzerverzeichnisse der Netzwerkbenutzer umfassende Verzeichnis muss per AFP freigegeben werden.

einem dynamischen AFP-Automount, muss diese Angabe um ein weiteres benutzerspezifisches Attribut ergänzt werden, welches den Servernamen und den Namen der Freigabe beinhaltet.

Das hängt damit zusammen, dass in diesem Fall die automatische Aktivierung mit Gast-Rechten erfolgt (vgl. oben die Festlegung der AFP-Authentifizierungsmethode beim Automount-Eintrag als `AUTH=NO%20USER%20AUTHENT`), so dass beim Anmelden im Allgemeinen eine weitere Authentifizierung und Aktivierung notwendig ist, um das Benutzerverzeichnis überhaupt verwenden zu können.

Der Servername und der Name der Freigabe wird dann auch mit Hilfe des zusätzlichen Attributs `home_loc` spezifiziert. Der Wert des Attributs ist in einer XML-ähnlichen Notation strukturiert und hat die Form:

```
<home_dir>
<url>
afp://server/freigabe
</url>
<path>
benutzerverzeichnis
</path>
</home_dir>
```

Ein gültiger Wert kann demnach z. B. lauten (Abb. 4.10):

```
$ nicl / -read /users/test home_loc
home_loc: < home_dir ><url>afp://server.example.com/
          Users</url><path>test</path></home_dir >
```

Um eine zentrale Benutzerverwaltung mit zentral gespeicherten Benutzerverzeichnissen zu realisieren, sind demnach die folgenden Schritte notwendig:

- Erstellung eines zentralen Verzeichnisses für die Benutzerverzeichnisse auf einem geeigneten Server-Rechner, z. B. `/Shares/Users`
- Erstellung der individuellen Benutzerverzeichnisse innerhalb dieses Verzeichnisses, also z. B. `/Shares/Users/test` u. a.
- Freigabe des zentralen Verzeichnisses mittels AFP oder NFS
- Erstellung eines entsprechenden Automount-Eintrags im zentralen Verzeichnisdienst (hier NetInfo)
- Änderung des Attributs `home` und ggf. Ergänzung des Attributs `home_loc` bei allen im zentralen Verzeichnisdienst eingetragenen Benutzern.

Dabei spielt es keine Rolle, ob der zentrale Verzeichnisdienst (in unserem Fall die NetInfo-Datenbank `network`) und die Freigabe mit dem zentralen Benutzerverzeichnis auf ein und demselben Rechner realisiert werden oder auf zwei unterschiedlichen Servern.

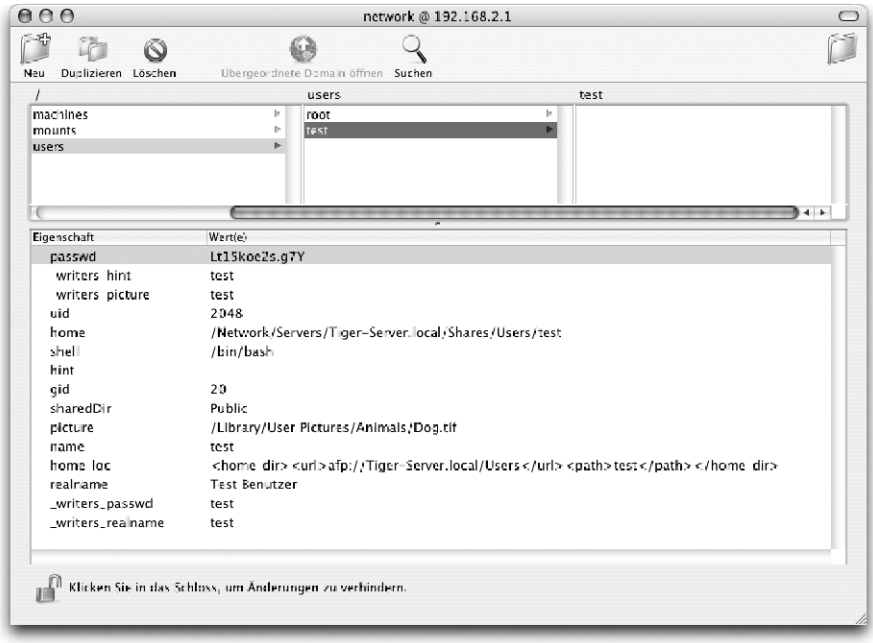


Abb. 4.10. Die Attribute home und home_loc (nur bei AFP-Freigaben) müssen angepasst bzw. ergänzt werden.

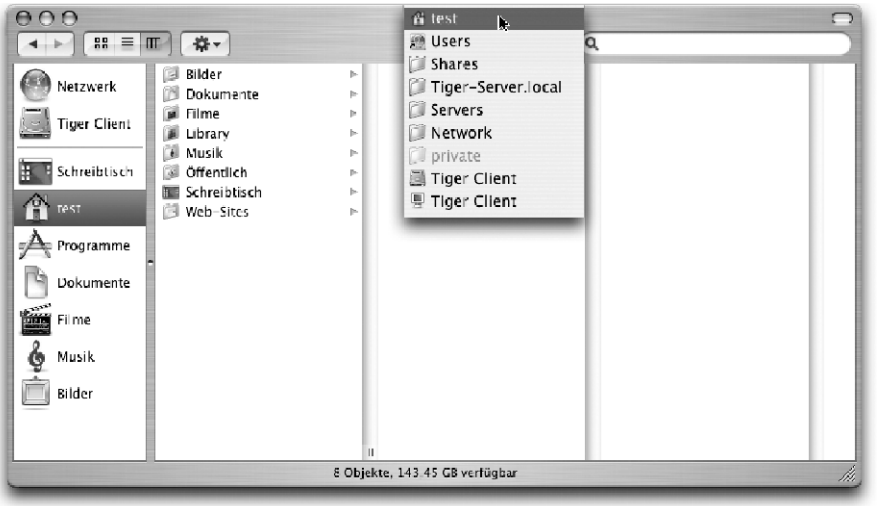


Abb. 4.11. Wenn alles geklappt hat, kann man sich von jedem beliebigen Client-Rechner aus anmelden und arbeitet mit einem zentralen Benutzerverzeichnis.

In jedem Fall ist nach der hier gezeigten Konfiguration für die zentral eingerichteten Benutzer die Anmeldung auf beliebigen, an den zentralen Verzeichnisdienst gebundenen Client-Rechnern möglich.

4.2 LDAP

Um die Interoperabilität zwischen Verzeichnisdatenbanken unterschiedlichen Ursprungs und unterschiedlichster Bauart gewährleisten zu können, bedarf es eines gemeinsamen Zugriffsprotokolls.

Heutzutage setzt sich hierfür immer mehr das *Lightweight Directory Access Protocol*[6] (kurz LDAP) durch. Die aktuelle Version 3 des LDAP-Standards (LDAPv3) wird dabei durch die RFCs 2251-2256, 2829-2830 und 3377 definiert.

Ein typisches LDAP-System besteht aus den drei Komponenten LDAP-Client, LDAP-Server und Verzeichnisdatenbank (LDAP-Backend):

- Der *LDAP-Client* greift mit Hilfe des LDAP-Protokolls auf die in der Verzeichnisdatenbank abgelegten Daten zu. LDAP-Clients sind oft Bestandteile von Anwendungsprogrammen. Unter Mac OS X verfügen z. B. die Anwendungen Adressbuch und Mail über einen integrierten LDAP-Client. Darüberhinaus verfügt OpenDirectory über ein LDAP-Modul, mit dessen Hilfe die in einer über LDAP zugänglichen Verzeichnisdatenbank

[6] frei übersetzt: leichtgewichtiges Zugriffsprotokoll für Verzeichnisdatenbanken

abgelegten Benutzerdaten für die Authentifizierung verwendet werden können.
- Der *LDAP-Server* stellt eine standardisierte Anfragestelle zur Verfügung, über die Lese- und Schreibzugriffe auf die in der Verzeichnisdatenbank gespeicherten Daten erfolgen können.
- Die *Verzeichnisdatenbank* dient schließlich der (persistenten) Speicherung der Verzeichnisdaten. Als Datenbank kann dabei, zumindest theoretisch, eine beliebige Datenquelle dienen. Mac OS X enthält (seit der Version 10.2) beispielsweise einen Adapter, die so genannte *NetInfo Bridge*, mit dessen Hilfe die in einer NetInfo-Datenbank gespeicherten Verzeichnisdaten über den ebenfalls mitgelieferten LDAP-Server veröffentlicht werden können.

Mac OS X wird zusammen mit OpenLDAP 2 ausgeliefert. Wie viele andere Komponenten von Mac OS X ist auch OpenLDAP das Ergebnis eines Open-Source-Projekts und für viele andere Plattformen (u. a. Solaris, Linux, FreeBSD und Windows) verfügbar. OpenLDAP 2 unterstützt LDAPv3 und baut auf der Referenzimplementierung von LDAPv2 auf, die ursprünglich an der University of Michigan entwickelt wurde. Zum Lieferumfang von Mac OS X 10.4.2 gehört die Version 2.2.19 von OpenLDAP:

```
$ /usr/libexec/slapd -V
@(#) $OpenLDAP: slapd 2.2.19 $
```

4.2.1 Das LDAP-Datenmodell

Ein LDAP-Server präsentiert die in der Verzeichnisdatenbank abgelegten Daten als eine Hierarchie von LDAP-Objekten in einer Baumstruktur, dem *Data Information Tree* (kurz DIT). Jedes LDAP-Objekt entspricht dabei genau einem abstrakten Datensatz – z. B. einem Personendatensatz – in der Verzeichnisdatenbank. LDAP-Objekte werden wie üblich mit Hilfe von Attributen charakterisiert, wobei ein Attribut grundsätzlich auch mehrere Werte aufnehmen kann.

Anders als z. B. NetInfo-Datensätze sind LDAP-Datensätze immer typisiert, d. h. jedes Objekt hat einen bestimmten Typ, die so genannte *Objekt-Klasse*, welche die erlaubten Attribute definiert. Die Objekt-Klasse legt dabei nicht nur den Wertebereich der Attribute fest, sondern definiert auch für jedes Attribut Regeln für die Durchführung von Vergleichsoperationen. Außerdem wird in der Objekt-Klasse festgelegt, ob ein Attribut lediglich erlaubt ist, und u. U. weggelassen werden kann, oder im Gegenteil zwingend vorgeschrieben ist. Ebenso lässt sich festlegen, ob ein Attribut immer nur einen, oder manchmal auch mehrere Werte aufnehmen darf.

Der Typ eines Objekts wird mit Hilfe des Attributs `objectClass` angegeben. Dieses Attribut muss natürlich mindestens einen Wert besitzen, darf aber auch mehrere Werte aufnehmen. Ein Beispiel für eine für Personendatensätze gebräuchliche Klasse ist `inetOrgPerson`.

Nicht jede Klasse darf mit jeder anderen kombiniert werden. Der LDAP-Standard unterscheidet in diesem Zusammenhang drei Kategorien von Klassen: *Structural*, *Auxiliary* und *Abstract Object Classes*. Jedes Objekt muss dabei eindeutig von einer einzigen *Structural Object Class* abstammen. Hat man eine solche Klasse eindeutig definiert, dürfen zusätzlich eine oder mehrere *Auxiliary Object Classes* angegeben werden, um weitere Attribute zuzulassen. *Abstract Object Classes* sind schließlich Hilfskonstrukte, die bei der Definition von neuen Klassen verwendet werden können und bei der Erstellung von Objekten bzw. Datensätzen eine untergeordnete Rolle spielen.

Eine Besonderheit von LDAP ist die Art und Weise, wie Objekte identifiziert werden. Um Objekte innerhalb des DIT eindeutig identifizieren zu können, wird in LDAP ein so genannter *Distinguished Name* (kurz DN) verwendet. Ein DN ist mit einem absoluten Pfad in einem Dateisystem vergleichbar und besteht aus einer mit Kommas getrennten Liste von so genannten *Relative Distinguished Names* (kurz RDNs).

Ein RDN identifiziert dabei eindeutig ein Objekt innerhalb einer Hierarchieebene. Anders als bei Dateisystemen (oder auch z. B. NetInfo) üblich, muss man als RDN neben dem Wert auch den Attributnamen angeben. Der Grund: in Dateisystemen sind alle Objekte/Dateien – zumindest was grundlegende Dateioperationen anbetrifft – typengleich. Insbesondere verfügen alle Dateien in einem Dateisystem über das Attribut „Name“, das auf jeder Hierarchieebene (Verzeichnis) eindeutig ist. Anders die Objekte in einem LDAP DIT. Hier sind die Objekte im Allgemeinen von unterschiedlichem Typ mit unterschiedlichen Attributen: die Verfügbarkeit eines globalen Attributs, das dazu auch noch das Eindeutigkeitskriterium erfüllt, kann nicht garantiert werden.

Man muss sich demnach auf jeder Hierarchieebene mit der Angabe des jeweils eindeutigen Attributs behelfen, so dass ein typischer DN, etwa für einen Personendatensatz, die Form haben könnte:

```
cn=Rafael Kobylinski,ou=people,dc=example,dc=com
```

Dabei kann es natürlich Fälle geben, in denen innerhalb einer Hierarchieebene keines der Attribute für sich genommen eindeutig ist. Ein einfaches Beispiel dafür wären Datensätze von zwei Personen mit den gleichen Vor- und Nachnamen. Um die Identifizierung von Datensätzen auch in solchen Fällen gewährleisten zu können, erlaubt LDAP RDNs mit mehr als einem Attribut, z. B.:

```
cn=Rafael Kobylinski+ou=Hauptniederlassung,ou=people,
dc=example,dc=com
```

4.2.2 LDAP-Server aktivieren

Der OpenLDAP-LDAP-Server `slapd` ist unter Mac OS X 10.4 für den Einsatz mit der NetInfo-Bridge vorkonfiguriert. Unter Mac OS X 10.3 sind

die Konfigurationdaten von `slapd` dabei in der Standardkonfigurationsdatei `/etc/openldap/slapd.conf` bereits enthalten:

```
$ sudo cat /etc/openldap/slapd.conf
##
# slapd.conf file for NetInfo bridge
##

include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/samba.schema
include         /etc/openldap/schema/apple.schema
pidfile         /var/run/slapd.pid
argsfile        /var/run/slapd.args
allows          bind_v2
schemacheck     off

database        netinfo
suffix          ""
flags           DSENGINE_FLAGS_NATIVE_AUTHORIZATION
                DSSTORE_FLAGS_ACCESS_READWRITE
datasource      /var/db/netinfo/network.nidb
include         /etc/openldap/schema/netinfo.schema
```

In Mac OS X 10.4 gibt es nur die Datei `/etc/openldap/slapd.conf.default`, die eine Kopie der Konfigurationsdaten enthält. Da `slapd` seine Konfiguration nach dem Start in der Datei `slapd.conf` sucht, muss man unter Mac OS X 10.4 entweder die Datei kopieren oder `slapd` zusammen mit dem Parameter `-f` starten. Über den Parameter `-f` kann man den Pfad einer alternativen Konfigurationsdatei angeben.

```
$ sudo cat /etc/openldap/slapd.conf.default
Password:
##
# slapd.conf file for NetInfo bridge
##

include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/misc.schema
include         /etc/openldap/schema/samba.schema
include         /etc/openldap/schema/apple.schema
pidfile         /var/run/slapd.pid
argsfile        /var/run/slapd.args
allows          bind_v2
schemacheck     off

database        netinfo
suffix          ""
flags           DSENGINE_FLAGS_NATIVE_AUTHORIZATION
datasource      /var/db/netinfo/network.nidb
include         /etc/openldap/schema/netinfo.schema
```

```
$ sudo cp /etc/openldap/slapd.conf.default
          /etc/openldap/slapd.conf
```

Wie üblich werden innerhalb einer `slapd`-Konfigurationsdatei alle mit einem `#` beginnenden Zeilen als Kommentare ignoriert. Beide angegebenen Konfigurationsdateien lassen sich in zwei Abschnitte unterteilen: im ersten, globalen Konfigurationsabschnitt werden Festlegungen für den LDAP-Server als Ganzes getroffen, während im zweiten, datenbankspezifischen Abschnitt Festlegungen stehen, die nur eine bestimmte Verzeichnisdatenbank betreffen. Grundsätzlich sind mehrere datanbankspezifische Abschnitte möglich, obgleich die von Apple ausgelieferten Konfigurationsdateien nur einen solchen Abschnitt enthalten.

Im globalen Abschnitt werden zunächst mit Hilfe der `include`-Anweisungen Konfigurationsdateien mit Objekt- und Attributdefinitionen geladen. Danach wird mit Hilfe der Anweisungen `pidfile` und `argsfile` festgelegt, wo zur Laufzeit eine Datei mit der Prozessnummer und den beim Aufruf verwendeten Argumenten abgelegt werden soll.

Die `allows`-Anweisung schaltet den Kompatibilitätsmodus für ältere LDAPv2-Clients ein. Die `schemacheck`-Anweisung schaltet die Typüberprüfung aus – da NetInfo kein Typkonzept kennt, wurden NetInfo-Datenbanken von den Anwendern sehr häufig um eigene Attribute erweitert. Eine Typüberprüfung würde in diesen Fällen zu Fehlermeldungen führen.

Der datenbankspezifische Abschnitt beginnt mit der Anweisung `database` und endet erst mit dem Dateiende bzw. mit dem Auftreten einer weiteren `database`-Anweisung (im Beispiel findet sich nur ein einziger datanbankspezifischer Abschnitt).

Die Anweisung `database` legt die Art der Datenbank fest. Der Wert `netinfo` teilt dem Server mit, dass er für den Zugriff auf die Datenbank die NetInfo-Bridge verwenden soll bzw. dass es sich bei dem Backend um eine NetInfo-Datenbank handelt. Mit der Anweisung `datasource` wird der Pfad zur Datenbank angegeben und mit der Anweisung `include` eine weitere Konfigurationsdatei geladen, welche die genaue Abbildung der NetInfo-Attribute auf LDAP-Attribute enthält. Mit der Anweisung `flags` werden Parameter an die NetInfo-Bridge übergeben, die dafür sorgen, dass die Datenbank per LDAP zwar grundsätzlich schreibbar (Option `DSSTORE_FLAGS_ACCESS_READWRITE`)[7] ist, aber nur wenn die NetInfo-Zugriffsbeschränkungen das erlauben (Option `DSENGINE_FLAGS_NATIVE_AUTHORIZATION`).

Die Anweisung `suffix` ordnet die Verzeichnisdatenbank schließlich in ein Datenbanken- (und potentiell auch LDAP-Server-) übergreifendes Namensschema ein. Der Anweisung folgt normalerweise ein DN, der den Pfad zur Wurzel der Datenbank in einem solchen größeren Kontext repräsentiert. Normalerweise ist es relativ egal, welches Suffix man hier verwendet. Für Testzwecke kann man das Suffix auch durchaus erstmal leer lassen.

[7] Diese Option fehlt in der Mac-OS-X-10.4-Konfigurationsdatei.

Später verwendet man häufig eine von der DNS-Domain abgeleitete DN, z. B. dc = example, dc = com.

Der Serverprozess selbst wird mit Hilfe des Befehls `/usr/libexec/slapd` gestartet. Mit Hilfe der Option `-d` (gefolgt von einer numerisch anzugebenden Protokollierungsstufe) verbleibt der Prozess nach dem Start im Vordergrund und gibt zahlreiche Statusmeldungen aus. Nützlich ist z. B. die Protkollierungsstufe 256, die alle Suchanfragen und deren Parameter ausgibt.

Ein einfacher Aufruf dieses Befehls resultiert allerdings erstmal in einer Fehlermeldung:

```
$ /usr/libexec/slapd -d 256
daemon: bind(6) failed errno=13 (Permission denied)
daemon: bind(6) failed errno=13 (Permission denied)
slapd stopped.
connections_destroy: nothing to destroy.
```

Die Fehlermeldung wird durch den Versuch verursacht, den privilegierten TCP-Port 389 zu öffnen. Ein zweiter Aufruf, diesmal mit root-Rechten, führt allerdings meistens auch nicht sofort zum gewünschten Ergebnis:

```
$ sudo /usr/libexec/slapd -d 256
@(#) $OpenLDAP: slapd 2.2.19 $
bdb_back_initialize: Sleepycat Software: Berkeley DB 4.2.52:
                                          (December  3, 2003)
/etc/openldap/slapd.conf: line 20: could not open datasource
             "/var/db/netinfo/network.nidb" flags [00000400]
slapd stopped.
connections_destroy: nothing to destroy.
```

Hier wird die ausgewählte NetInfo-Datenbank mit der Bezeichnung „network“ schlichtweg deswegen nicht gefunden, weil sie in der Einzelplatzvariante normalerweise fehlt. Hat man eine Netzwerk-Datenbank manuell angelegt, lässt sich `slapd` allerdings auch auf der Einzelplatzversion von Mac OS X mit der mitgelieferten Konfigurationsdatei starten, und greift über die NetInfo-Bridge auf diese Datenbank zu:

```
$ sudo /usr/libexec/slapd -d 256
@(#) $OpenLDAP: slapd 2.2.19 $
bdb_back_initialize: Sleepycat Software: Berkeley DB 4.2.52:
                     (December  3, 2003)
schemamap_add_oc: Could not add objectClass mapping for
                  directory /computers: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute realname: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute comment: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute en_address: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute groups: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute uid: Invalid Path
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute gid: Invalid Path
```

```
schemamap_add_at: Could not add attribute mapping for
                  NetInfo attribute
                  authentication_authority: Invalid Path
...
slapd starting
```

Man sollte sich nicht durch die zahlreichen Fehlermeldungen abschrecken lassen: trotz dieser Meldungen läuft der LDAP-Server und kann ohne weitere Konfiguration lokal[8] benutzt werden.

Ein auf diese Weise gestarteter LDAP-Server lässt sich am einfachsten mit Ctrl-C beenden:

```
...
slapd shutdown: waiting for 0 threads to terminate
slapd stopped.
```

Dabei wird die Verzeichnisdatenbank sauber geschlossen. Um die Datenintegrität zu gewährleisten, sollte man einen LDAP-Server ohne angeschlossenes Terminal ansonsten ausschließlich mit Hilfe des HUP-Signals beenden, z. B. mit:

```
$ sudo killall -HUP slapd
```

Natürlich wird man den LDAP-Server nur in der Experimentierphase manuell starten wollen. Um unter Mac OS X 10.3 automatisch beim Systemstart den LDAP-Server zu aktivieren, genügte es, in der Datei `/etc/hostconfig` den Eintrag `LDAPSERVER:=-YES-` zu ergänzen. Diese kleine Änderung war ausreichend – der Wert der Variablen `LDAPSERVER` wurde bereits im Auslieferungszustand vom StartupItem LDAP abgefragt:

```
$ cat /System/Library/StartupItems/LDAP/LDAP
#!/bin/sh

##
# LDAP server
##

. /etc/rc.common

StartService ()
{
    ##
    # Start up LDAP server
    ##
    if [ "${LDAPSERVER:=-NO-}" = "-YES-" ]; then
        if ! pid=$(GetPID slapd); then
            ConsoleMessage "Starting LDAP server"
            if [ "${LDAPSSL:=-NO-}" = "-YES-" ]; then
                slapd -h "ldap:// ldaps:///"
                        else
                slapd
            fi
```

[8] Der Zugriff über das Netzwerk wird aufgrund der NetInfo-Zugriffsbeschränkungen von der NetInfo-Bridge zunächst blockiert.

```
        fi
    fi
...
```

In der Einzelplatzvariante von Mac OS X 10.4 fehlt dieses StartupItem – es wird dort ja normalerweise nicht benötigt. Um den LDAP-Server dennoch automatisch zu starten, muss man demnach ein eigenes StartupItem (in /Library/StartupItem/) oder eine eigene `launchd`-Konfigurationsdatei (in /Library/LaunchDaemons/) erstellen.

4.2.3 LDAP-Verzeichnisse durchsuchen

Nachdem der LDAP-Server erfolgreich gestartet wurde, sollte man den Inhalt der Verzeichnisdatenbank überprüfen. Am einfachsten geht das durch das Absetzen einer einfachen LDAP-Suchanfrage.

Der Befehl `ldapsearch` ist ein einfaches Werkzeug, mit dem sich LDAP-Suchanfragen formulieren lassen. Mit `ldapsearch` können wir verifizieren, ob der LDAP-Server tatsächlich den Inhalt der NetInfo-Datenbank zugänglich macht:

```
$ ldapsearch -x -h 127.0.0.1 -s one
# extended LDIF
#
# LDAPv3
# base <> with scope one
# filter: (objectclass=*)
# requesting: ALL
#

# machines
dn: cn=machines
cn: machines
objectClass: container

# users
dn: cn=users
cn: users
objectClass: container

# mounts
dn: cn=mounts
cn: mounts
objectClass: container

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Der obige Aufruf von `ldapsearch` liefert im Wesentlichen die Verzeichnisnamen der ersten Hierarchieebene in der NetInfo-Datenbank und verwendet dazu die folgenden Optionen:

-x authentifiziert die Suchanfrage mit Hilfe eines einfachen Authentifizierungsmechanismus namens *LDAP-Bind*.

-h 127.0.0.1 gibt den Hostnamen bzw. die IP-Adresse des zu kontaktierenden LDAP-Servers vor. Die Angabe der Loopback-Adresse ist hier normalerweise redundant, da `ldapsearch` den Hostnamen localhost als Voreinstellung verwendet. Bei Verwendung der NetInfo-Bridge muss die IPv4-Loopback-Adresse jedoch explizit angegeben werden, da `ldapsearch` ansonsten versucht, eine Verbindung über die IPv6-Loopback-Adresse aufzubauen, was mit der NetInfo-Bridge nicht funktioniert.

-s one begrenzt die „Reichweite“ (engl. *scope*) der Suche auf eine Hierarchieebene. Die Voreinstellung ist `-s sub`, was der Suche im gesamten Verzeichnisunterbaum entspricht. Wird dieser Parameter weggelassen, werden folgerichtig alle in der NetInfo-Datenbank enthaltenen Objekte ausgegeben.

Allgemein kommen meistens mindestens drei weitere (Arten von) Optionen dazu:

- ein Basisobjekt (engl. *search base*) für die Suche. Das Basisobjekt dient als Ausgangspunkt für die Suche, so dass nur Objekte unterhalb dieses Objekts durchsucht werden. Der DN des Basisobjekts kann mit Hilfe der Option `-b` angegeben werden.
- ein Suchfilter nach RFC 2254. Ein solcher Suchfilter ist ein Ausdruck, der normalerweise durch einfache Klammern begrenzt wird und aus einem Attributnamen, einem Vergleichsoperator und einem Attributwert besteht. Als Vergleichsoperatoren sind u. a. `=` (Gleichheit) und `<=` bzw. `>=` (Vergleichsoperatoren/) zugelassen. Innerhalb der Werte kann der Stern als Platzhalter für Null oder mehrere beliebige Zeichen verwendet werden. Ein einfacher Suchfilter könnte demnach lauten: `(uid=rkk)` oder `(cn=Rafael*)`. Suchfilter lassen sich mit Hilfe der Boole'schen Operatoren `&` (logisches UND), `|` (logisches ODER) sowie `!` (logisches NICHT) beliebig kombinieren, wobei eine Prefix-Notation verwendet wird: z. B. `(& (objectClass=inetOrgPerson) (uid=rkk))`.
- eine Liste von Attributen, die für jeden gefundenen Eintrag auszugeben sind. Fehlt diese Liste, werden einfach alle Attribute ausgegeben[9].

Um in der gesamten Datenbank nach Einträgen vom Typ `inetOrgPerson` zu suchen und deren Attribute `uidNumber`, `uid` und `loginShell` auszugeben, müsste man demnach die Suchanfrage wie folgt formulieren:

```
$ ldapsearch -x -h 127.0.0.1 "(objectClass=inetOrgPerson)"
                    uidNumber uid loginShell
# extended LDIF
#
```

[9] Einige interne Attribute werden nur ausgegeben, wenn sie explizit angegeben worden sind, oder wenn die Liste der auszugebenden Attribute ein Plus-Zeichen enthält.

```
# LDAPv3
# base <> with scope sub
# filter: (objectClass=inetOrgPerson)
# requesting: uidNumber uid loginShell
#

# root, users
dn: uid=root,cn=users
uid: root
uidNumber: 0

# test, users
dn: uid=test,cn=users
uidNumber: 2048
loginShell: /bin/bash
uid: test

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Da in der NetInfo-Datenbank alle Benutzereinträge im Verzeichnis `/users/` abgelegt sind, kann man das gleiche Ergebnis erzielen, wenn man das Basisobjekt der Suche entsprechend setzt und beliebige Objekte zulässt:

```
$ ldapsearch -x -h 127.0.0.1 -b "cn=users" "(objectClass=*)"
                    uidNumber uid loginShell
# extended LDIF
#
# LDAPv3
# base <cn=users> with scope sub
# filter: (objectClass=*)
# requesting: uidNumber uid loginShell
#

# users
dn: cn=users

# root, users
dn: uid=root,cn=users
uid: root
uidNumber: 0

# test, users
dn: uid=test,cn=users
uidNumber: 2048
loginShell: /bin/bash
uid: test

# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Sofern der LDAP-Server mit der Option `-d 256` gestartet wurde, lassen sich anhand der Ausgaben von `slapd` die erhaltenen Suchanfragen leicht nachvollziehen:

```
...
conn=0 fd=10 ACCEPT from IP=127.0.0.1:49174 (IP=0.0.0.0:389)
conn=0 op=0 BIND dn="" method=128
conn=0 op=0 RESULT tag=97 err=0 text=
conn=0 op=1 SRCH base="" scope=1 deref=0
            filter="(objectClass=*)"
conn=0 op=1 SEARCH RESULT tag=101 err=0 nentries=3 text=
conn=0 op=2 UNBIND
conn=0 fd=10 closed
...
...
conn=1 fd=10 ACCEPT from IP=127.0.0.1:49175 (IP=0.0.0.0:389)
conn=1 op=0 BIND dn="" method=128
conn=1 op=0 RESULT tag=97 err=0 text=
conn=1 op=1 SRCH base="" scope=2 deref=0
            filter="(objectClass=inetOrgPerson)"
conn=1 op=1 SRCH attr=uidNumber uid loginShell
conn=1 op=1 SEARCH RESULT tag=101 err=0 nentries=2 text=
conn=1 op=2 UNBIND
conn=1 fd=10 closed
...
...
conn=2 fd=10 ACCEPT from IP=127.0.0.1:49176 (IP=0.0.0.0:389)
conn=2 op=0 BIND dn="" method=128
conn=2 op=0 RESULT tag=97 err=0 text=
conn=2 op=1 SRCH base="cn=users" scope=2 deref=0
            filter="(objectClass=*)"
conn=2 op=1 SRCH attr=uidNumber uid loginShell
conn=2 op=1 SEARCH RESULT tag=101 err=0 nentries=3 text=
conn=2 op=2 UNBIND
conn=2 fd=10 closed
...
```

4.2.4 Graphische LDAP-Werkzeuge

Obwohl man die Datenbestände eines LDAP-Servers ohne weiteres ausschließlich mit `ldapsearch` durchsuchen kann, ist eine graphische Oberfläche weitaus bequemer.

Der frei erhältliche *Java LDAP Browser/Editor* von Jaroslaw Gawor ist eine plattformunabhängige Java-Applikation, die auch unter Mac OS X verhältnismäßig gut läuft und daher häufig empfohlen wird:

`http://www-unix.mcs.anl.gov/~gawor/ldap/`

Um mit Hilfe dieser Applikation eine Verbindung zu unserem weiter oben aktivierten LDAP-Server aufbauen zu können, sollte man am besten ein so genanntes Session-Profil einrichten.

Das Programm ist weitgehend selbsterklärend. Der Session-Profil-Editor wird automatisch beim Aufruf des Befehls „Connect...“ aus dem File-Menü aktiviert. Für den Beispiel-LDAP-Server sind dort folgende Verbindungsparameter einzustellen (Abb. 4.12):

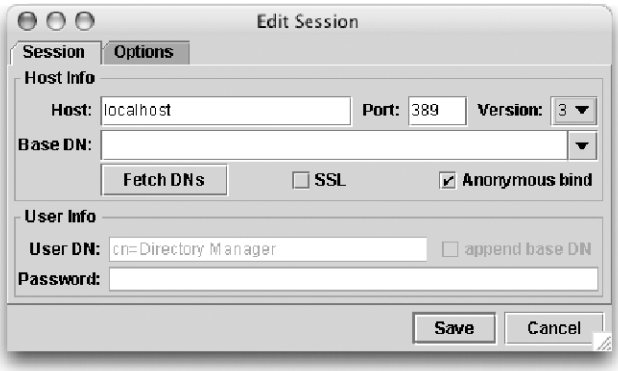


Abb. 4.12. Die Verbindungsparameter müssen im Session-Profil-Editor eingestellt werden.

- Host: localhost. Im Gegensatz zu `ldapsearch` verwendet der LDAP Browser/Editor ausschließlich IPv4 für den Verbindungsaufbau. Natürlich kann man auch direkt die Loopback-Adresse 127.0.0.1 angeben.
- Base DN muss leer bleiben, solange kein „Suffix“ für die NetInfo-Datenbank definiert worden ist.
- Port: 389 (entspricht der Standardeinstellung)

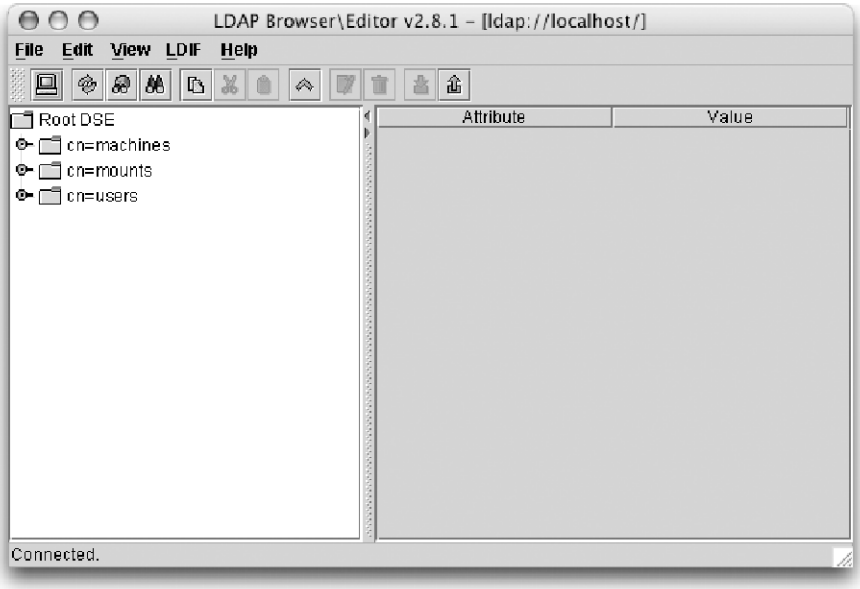


Abb. 4.13. Nach erfolgreicher Anmeldung zeigt der LDAP Browser/Editor die oberste Hierarchieebene der Verzeichnisdatenbank.

- Version kann sowohl den Wert 3 (Standardeinstellung) als auch den Wert 2 annehmen, sofern der LDAP-Server für den Betrieb im LDAPv2-Kompatibilitätsmodus konfiguriert wurde (siehe slapd.conf).
- SSL sollte deaktiviert bleiben
- Anonymous bind sollte man hingegen aktivieren (entspricht der Option `-x` von `ldapsearch`).

Nach erfolgreicher Anmeldung zeigt der LDAP Browser/Editor die oberste Hierarchieebene der Verzeichnisdatenbank (Abb. 4.13). Mit `/machines`, `/mounts` und `/users` findet man dort (wie erwartet) exakt die gleichen Verzeichnisse wie auf der obersten Hierarchieebene der verwendeten NetInfo-Datenbank (Abb. 4.15). Die NetInfo-Bridge übernimmt die Abbildung zwischen LDAP und NetInfo in Echtzeit.

Alternativ kann man neben dem auch auf anderen Plattformen sehr verbreiteten LDAP Browser/Editor auch die Mac-OS-X-spezifischen Anwendungen LDAPManager und LDAPper verwenden.

```
http://sourceforge.net/projects/ldapmanager/
http://carl-bell-2.baylor.edu/~Carl_Bell/stuff.html
```

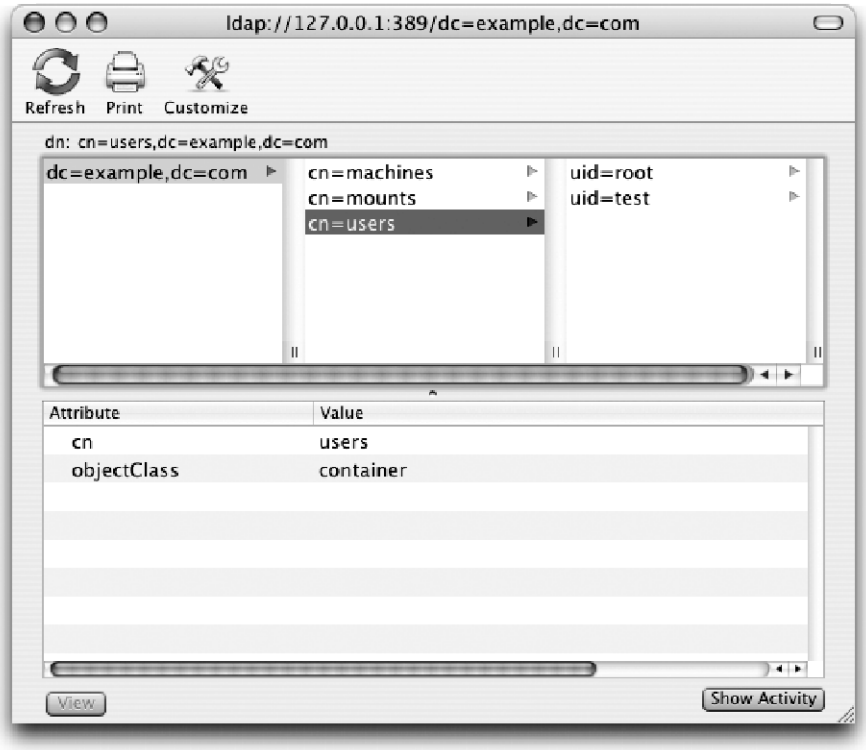


Abb. 4.14. Das Darstellungsfenster im LDAPManager ist dem NetInfo Manager nachempfunden.

Abb. 4.15. Die Verzeichnisse auf der obersten Hierarchieebene der lokalen NetInfo-Datenbank sind beim Einsatz der NetInfo-Bridge natürlich die gleichen wie die, die über LDAP sichtbar sind.

Der Funktionsumfang dieser Anwendungen ist gegenüber dem LDAP Browser/Editor etwas eingeschränkt. LDAPManager verfügt über eine dem NetInfo-Manager nachempfundene Darstellung der Verzeichnisdatenbank (Abb. 4.14), während LDAPper die Elemente der Datenbank in einer klassischen Baumstruktur anzeigt (Abb. 4.16).

4.2.5 Das LDAP-Suffix der NetInfo-Datenbank ändern

Bisher haben wir als LDAP-Suffix, also als den Pfad zum Wurzelobjekt der (NetInfo-)Datenbank, die leere Zeichenkette angegeben. Um eine spätere Zusammenführung von LDAP-Datenbanken zu ermöglichen, hat es sich heutzutage allerdings eingebürgert, jeder LDAP-Datenbank ein eindeutiges Suffix zuzuteilen.

Um die weltweite Eindeutigkeit zu gewährleisten, wird dieses Suffix in der Regel vom bereits vorhandenen DNS-Domainnamen abgeleitet[10]. Jede

[10] Details zu dieser Abbildung sind im RFC 2247 dokumentiert.

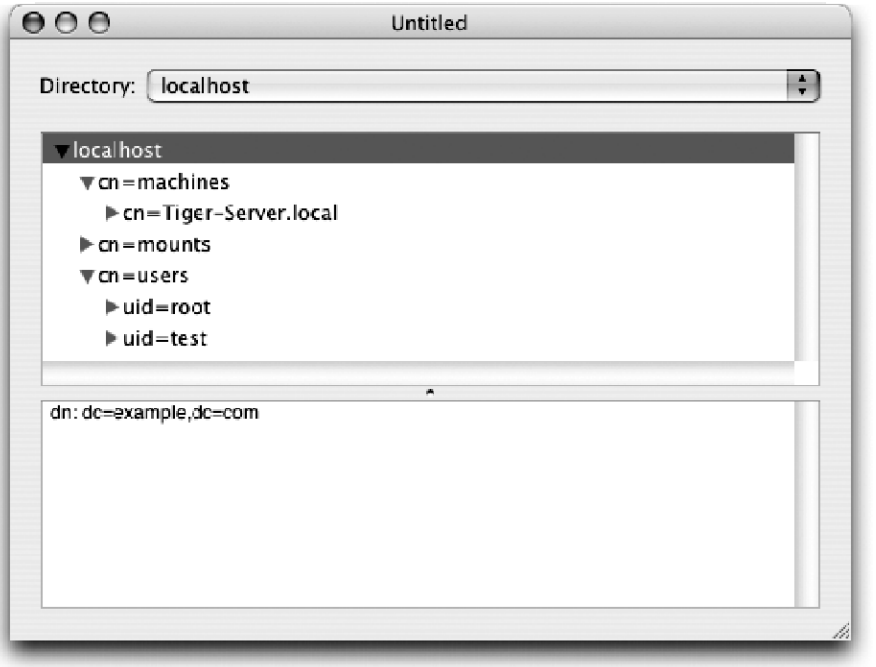


Abb. 4.16. LDAPper verwendet eine klassische Baumstruktur, um die Elemente der Verzeichnisdatenbank anzuzeigen.

Komponente des DNS-Domainnamens wird dabei auf ein Attribut vom Typ *Domain Name Component* (kurz *dc*) abgebildet, so dass z. B. dem Domainnamen „example.com“ der DN „dc=example,dc=com“ entspricht.

Der für die Verzeichnisdatenbank gültige Suffix wird in der Konfigurationsdatei des LDAP-Servers `slapd.conf` mit Hilfe der Anweisung `suffix` angegeben. Um den DN „dc=example,dc=com“ als Suffix zu verwenden, muss man demnach dort die Anweisung

```
suffix          ""
```

durch

```
suffix          "dc=example,dc=com"
```

ersetzen.

Für sich genommen reicht diese Änderung jedoch nicht aus: auch die NetInfo-Bridge muss über das Suffix informiert werden. Dazu muss in der NetInfo-Datenbank der Eintrag `/machines/localhost` um das Attribut Suffix ergänzt werden, also z. B.:

```
$ nicl . -read /machines/localhost
ip_address: 127.0.0.1
name: localhost
serves: ./local
```

```
$ sudo nicl . -create /machines/localhost suffix
  "dc=example,dc=com"
$ nicl . -read /machines/localhost
ip_address: 127.0.0.1
name: localhost
serves: ./local
suffix: dc=example,dc=com
```

Erst nach dieser zweiten Änderung und einem Neustart des LDAP-Servers kann die Verzeichnisdatenbank mit Hilfe des neuen Suffix angesprochen werden (vgl. auch Abb. 4.17):

```
$ ldapsearch -x -h 127.0.0.1 -s one -b "dc=example,dc=com"
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope one
# filter: (objectclass=*)
# requesting: ALL
#

# machines, example.com
dn: cn=machines,dc=example,dc=com
cn: machines
objectClass: container

# users, example.com
dn: cn=users,dc=example,dc=com
cn: users
objectClass: container

# mounts, example.com
dn: cn=mounts,dc=example,dc=com
cn: mounts
objectClass: container
```

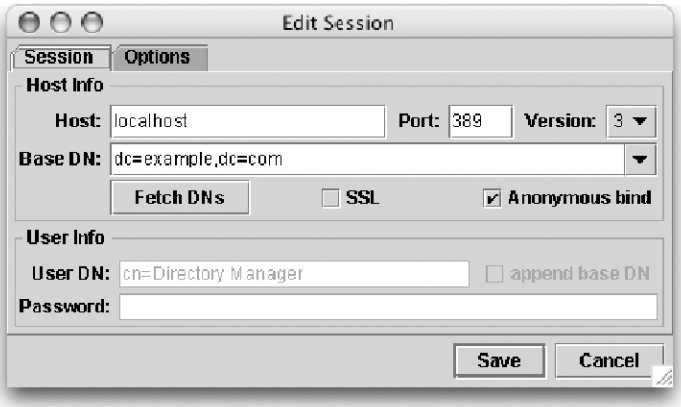


Abb. 4.17. Um die Änderung des LDAP-Suffix in der LDAP-Browser/Editor-Applikation nachzuvollziehen, muss man die „Base DN“ in den Verbindungseinstellungen ändern.

```
# search result
search: 2
result: 0 Success

# numResponses: 4
# numEntries: 3
```

Umgekehrt lässt sich die gleiche Datenbank nach dieser Änderung nicht mehr ohne die Angabe einer *Searchbase* (Option `-b` von `ldapsearch`) verwenden:

```
$ ldapsearch -x -h 127.0.0.1 -s one
# extended LDIF
#
# LDAPv3
# base <> with scope one
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 2
result: 32 No such object

# numResponses: 1
```

4.2.6 Ein zentrales LDAP-Adressbuch einrichten

Ein häufiges Einsatzgebiet von LDAP-Verzeichnissen sind unternehmensweite Kontaktverzeichnisse. Mit Hilfe eines solchen Verzeichnisses lassen sich verteilt gespeicherte Kontaktdaten von Personen konsolidieren, so dass Kontaktdaten von Mitarbeitern, Kunden, Lieferanten und anderen Kooperationspartnern bequem über eine einheitliche Schnittstelle abgefragt werden können.

Für den täglichen Einsatz sind die bisher vorgestellten Abfragemöglichkeiten natürlich zu kompliziert. Aus diesem Grunde geht man heutzutage immer mehr dazu über, rudimentäre, aber auf den jeweiligen Einsatzzweck angepasste LDAP-Abfragefunktionen direkt in Anwendungsprogramme zu integrieren.

Ein prominentes Beispiel für die Integration von LDAP sind die beiden mit Mac OS X ausgelieferten Programme „Adressbuch“ und „Mail“.

Mit Hilfe des Mac-OS-X-Adressbuchs lässt sich eine benutzerspezifische Kontaktdatenbank anlegen und pflegen, auf die grundsätzlich alle Mac-OS-X-Anwendungen Zugriff haben können. Das EMail-Programm Apple Mail verwaltet entsprechend keine eigene Kontaktdatenbank mit Personennamen und EMail-Adressen, sondern greift einfach auf die Adressbuch-Datenbank zu.

Das Adressbuch-Programm erlaubt darüber hinaus auch die Suche nach Kontaktinformationen in LDAP-Servern sowie die einfache Übertragung der so gefundenen Informationen in die lokale Datenbank mit Hilfe von Drag and Drop. In Apple Mail lässt sich wiederum einstellen, ob bei der Vervollständi-

gung von Email-Adressen LDAP-Server, die über das Adressbuchprogramm eingerichtet worden sind, automatisch mitdurchsucht werden sollen.

NetInfo als Kontaktdatenbank verwenden

Grundsätzlich lässt sich mit Hilfe der NetInfo-Bridge auch die lokale NetInfo-Datenbank nach Kontaktinformationen durchsuchen. Freilich ist diese Suche normalerweise recht erfolglos, weil die NetInfo-Benutzereinträge so gut wie keine Kontaktinformationen enthalten: die für die Adressbuch-Anwendung und Apple Mail relevanten Attribute fehlen einfach.

Natürlich lassen sich die fehlenden Attribute ohne weiteres ergänzen. Die Abfrageschnittstelle des Mac-OS-X-Adressbuchs versucht eine Reihe von Attributen zu lesen, von denen die meisten zum LDAP-Objekttyp inetOrgPerson gehören. Tabelle 4.2 listet die von der Addressbuch-Anwendung abgefragten Attribute zusammen mit den NetInfo-Attributen, auf die sie durch die NetInfo-Bridge abgebildet werden.

LDAP-Attribute, die die NetInfo-Bridge nicht kennt, werden einfach eins zu eins aus der NetInfo-Datenbank gelesen, so dass man z. B. in der NetInfo-Datenbank einfach ein Attribut `mail` ergänzen könnte, um Email-Adressen zu speichern.

Tabelle 4.2. Liste der vom Mac-OS-X-Adressbuch abgefragten LDAP-Attribute.

LDAP-Attribut	NetInfo-Attribut	Beschreibung
buildingName		Gebäudebezeichnung
c		Staat als ISO-Code, z. B. DE
cn	realname	„Common Name“
co		Staat als ausgeschriebener Name
destinationIndicator		
facsimileTelephoneNumber		Telefonnummer Fax
givenName	firstname	Vorname
jpegPhoto		Passphoto
l		Ortsbezeichnung
mail		Email-Adresse
mobile		Telefonnummer Mobiltelefon
o		Organisation/Firma
ou		Abteilung (Organizational Unit)
pager		Pager
postalCode	zip	Postleitzahl
sn	lastname	Nachname
st	stateOr-ProvinceName	Bundesstaat
street	address1	Straße und Hausnummer
telephoneNumber	phonenumber	Telefonnummer Festnetz
title		Funktionsbezeichnung/Akad. Grad

Um mit Hilfe des Mac-OS-X-Adressbuchs über LDAP auf die NetInfo-Datenbank zuzugreifen, muss man ansonsten in der Adressbuch-Einstellung die Kategorie LDAP wählen und ein passendes Verbindungsprofil erstellen.

Als Server muss man auch hier, ähnlich wie bei `ldapsearch`, die Adresse der Loopback-Schnittstelle (127.0.0.1) und nicht deren Namen (loopback) eingeben, da letztere zum erfolglosen Verbindungsaufbau mit IPv6 führt.

Als Suchbereich (entspricht dem Parameter `-b` von `ldapsearch`) muss das zuvor gewählte Suffix angegeben werden, also z. B. „dc=example,dc=com“. Um die Suche gleich auf das Verzeichnis `/users` zu beschränken (und damit etwas zu beschleunigen), kann man das Suffix um den RDN dieses Verzeichnisses („cn=users“) ergänzen (Abb. 4.18).

Als Name kann eine beliebige Bezeichnung gewählt werden. Die übrigen Voreinstellungen des Profils müssen nicht verändert werden.

Nach Abschluss der Konfiguration steht der LDAP-Server im Adressbuch innerhalb der Gruppe Verzeichnisse als Eintrag mit der im Profil eingestellten Bezeichnung zur Verfügung.

Da die Adressbuch-Anwendung die konfigurierten LDAP-Server nicht von Haus aus durchsucht, muss man die Gruppe der Verzeichnisse (oder einen spezifischen LDAP-Server) explizit auswählen, um eine LDAP-Suche durchführen zu können.

Abb. 4.18. Das LDAP-Verbindungsprofil der Adressbuch-Anwendung.

Die LDAP-Suchanfrage wird dabei erst abgesetzt, nachdem man mindestens zwei Zeichen in das Suchfeld der Anwendung eingegeben hat. Abbildung. 4.19 zeigt das Ergebnis einer Suche in der lokalen NetInfo-Datenbank.

Die Ergebnisliste zeigt in der ersten Spalte (Spaltentitel: „Name“) entweder den Wert des Attributs `cn`, oder die Kombination der Attribute `givenName` (Vorname) und `sn` (Nachname). In der zweiten Spalte kann der Benutzer zwischen Email-Adresse (Attribut `mail`), Telefonnummer (Attribut `telephoneNumber`) und der Adresse (Kombination der Attribute `street`, `postalCode`, `st` und `co`) wählen.

Hat man `slapd` mit der Option `-d 256` gestartet, kann man im Protokoll genau sehen, welche Art von Suchanfrage bearbeitet worden ist. Ensprechend kann man nach dem Absetzen der in Abb. 4.19 dargestellten Anfrage lesen:

```
...
conn=8 op=1 SRCH base="cn=users,dc=example,dc=com" scope=2
            deref=0 filter="(|(givenName=te*)(sn=te*)
                             (cn=te*)(mail=te*))"
conn=8 op=1 SRCH attr=givenName sn cn mail telephoneNumber
            facsimileTelephoneNumber o title ou buildingName
            street l st postalCode c jpegPhoto mobile co
            pager destinationIndicator labeledURI IMHandle
...
```

Demnach sucht das Adressbuch den Suchstring in den Vornamen, Nachnamen, den „Commmon Names“ und in der Email-Adresse. Eine äquivalente Suchanfrage lässt sich mit `ldapsearch` wie folgt formulieren:

```
$ ldapsearch -x -h 127.0.0.1 -b "cn=users,dc=example,dc=com"
             "(|(givenName=te*)(sn= te*)(cn= te*)(mail= te*))"
             givenName sn cn mail telephoneNumber
             facsimileTelephoneNumber o title ou buildingName
             street l st postalCode c jpegPhoto mobile co pager
             destinationIndicator labeledURI IMHandle
```

Abb. 4.19. Das Ergebnis einer LDAP-Suchanfrage in der lokalen NetInfo-Datenbank.

```
# extended LDIF
#
# LDAPv3
# base <cn=users,dc=example,dc=com> with scope sub
# filter: (|(givenName=te*)(sn= te*)(cn= te*)(mail= te*))
# requesting: givenName sn cn mail telephoneNumber
              facsimileTelephoneNumber o title ou buildingName
              street l st postalCode c jpegPhoto mobile co pager
              destinationIndicator labeledURI IMHandle
#

# test, users, example.com
dn: uid=test,cn=users,dc=example,dc=com
cn: Test Benutzer

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Erwartungsgemäß liefert die Suchanfrage ausschließlich das Attribut `cn`, weil die übrigen Attribute (bzw. die entsprechenden NetInfo-Attribute) in einer nicht modifizierten NetInfo-Datenbank fehlen. Entsprechend kann das Adressbuch nur den Namen und nicht auch noch die Email-Adresse, Telefonnummer oder die Anschrift anzeigen.

Separate Kontaktdatenbank einrichten

In den meisten Fällen macht es natürlich mehr Sinn, Kontaktdaten in einer separaten Datenbank zu speichern, als sie in der lokalen NetInfo-Datenbank abzulegen.

Um eine separate Kontaktdatenbank frisch einzurichten, kann man entweder die vorhandene LDAP-Konfigurationsdatei modifizieren, oder aber eine komplett neue Konfigurationsdatei erstellen. Da man den LDAP-Server mit Hilfe der Option `-f` anweisen kann, anstelle der Standardkonfigurationsdatei (`/etc/openldap/slapd.conf`) eine andere Konfigurationsdatei zu verwenden, kann man zwischen zwei (oder mehreren) Konfigurationsdateien sehr einfach umschalten.

Am Anfang der Überlegungen steht die Entscheidung, an welcher Stelle des lokalen Dateisystems die LDAP-Konfigurationsdatei, die Datenbank und andere Hilfsdateien abgelegt werden sollten. Für Übungs- und Testzwecke genügt hierfür ein Unterverzeichnis (z. B. `~/LDAPAdressbuch`) im Home des Benutzers, für den Produktionsbetrieb könnte hingegen ein Verzeichnis unterhalb von `/Library` erstellt werden.

Als nächstes muss man eine LDAP-Konfigurationsdatei erstellen. Eine minimale Datei sollte die folgenden Konfigurationsanweisungen beinhalten:

```
$ cat /Library/LDAPAdressbuch/slapd.conf
#
```

```
# slapd.conf für ein zentrales Adressbuch
#

#
# Klassen und Attribute
#

include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema

#
# Protokollierung
#

pidfile         /Library/LDAPAdressbuch/slapd.pid
argsfile        /Library/LDAPAdressbuch/slapd.args

#
# Datenbank
#

## Datenbanktyp
database        bdb

suffix          "dc=example,dc=com"

rootdn          "cn=admin,ou=people,dc=example,dc=com"
rootpw          {SSHA}a0L0mF7qk9tQvaYBY3H3E4qa80LFvVtO

directory       /Library/LDAPAdressbuch/Datenbank
```

Im ersten Abschnitt („Klassen und Attribute“) werden mit Hilfe der `include`-Anweisungen alle für die zukünftigen Datenbankeinträge relevanten Objekttypen (Klassen) und deren Attribute geladen. Die für Kontaktinformationen vorgesehene Klasse `inetOrgPerson` enthält dabei fast alle vom Adressbuch abgefragten Attribute. Die entsprechenden Definitionen von `inetOrgPerson` sind in der Datei `inetorgperson.schema` enthalten. Da LDAP-Klassen Attribute einfacherer Klassen erben können, müssen zusätzlich auch noch einige einfachere Klassen und Attribute aus den Dateien `core.schema` und `cosine.schema` geladen werden, da `inetOrgPerson` auf ihnen aufbaut.

Im zweiten Abschnitt („Protokollierung“), werden die Dateien angegeben, die zur Laufzeit die Prozessnummer und die Aufrufparameter des LDAP-Servers aufnehmen. Als Pfad und Dateiname sind hier beliebige Angaben erlaubt.

Im dritten und letzten Abschnitt („Datenbank“) werden schließlich die Angaben zur Datenbank gemacht, in der die Kontaktdaten gespeichert werden sollen. Anders als bisher wurde hier mit der Anweisung `database` nicht die NetInfo-Bridge, sondern mit `bdb` ein anderer Adapter angegeben. Der Adapter `bdb` erlaubt den Zugriff auf *Berkeley-DB 4*-Datenbanken, den derzeit verbreitetsten und empfohlenen Datenbanktyp für neue LDAP-Verzeichnisdatenbanken.

Wie üblich, wird anschließend mit Hilfe der Anweisung `suffix` ein DN für das Wurzelobjekt der Datenbank angegeben.

Die darauf folgenden zwei Anweisungen `rootdn` und `rootpw` legen den Namen und das Passwort eines Benutzers mit universellen Schreib- und Leserechten fest. Die Anweisung `rootdn` legt dabei den Namen dieses Benutzers als DN und die Anweisung `rootpw` sein Passwort fest.

Das Passwort kann im Klartext oder als Hash angegeben werden. Die Art des Hashes bzw. der verwendete Algorithmus wird dabei in geschweiften Klammern vor den Hash gestellt. Um den Hash für ein beliebiges Passwort zu berechnen, kann der Befehl `slappasswd` verwendet werden:

```
$ slappasswd
New password:
Re-enter new password:
{SSHA}a0L0mF7qk9tQvaYBY3H3E4qa80LFvVtO
```

Mit der Anweisung `directory` wird schließlich der Speicherort für die Verzeichnisdatenbank angegeben. Ist das angegebene Verzeichnis leer, erzeugt `slapd` beim ersten Start eine leere Datenbank.

Optional können ein oder mehrere Attribute angegeben werden, für die ein Index angelegt werden soll. Ein Index kann gerade bei größeren Datenbeständen die Suche erheblich beschleunigen. Es wird normalerweise empfohlen, zumindest das `objectClass`-Attribut mit einem Index auszustatten. Um darüber hinaus auch noch einen Index für die von der Adressbuch-Anwendung verwendeten Suchattribute anzulegen, müsste die Konfigurationsdatei um folgende `index`-Anweisungen ergänzt werden:

```
$ cat /Library/LDAPAdressbuch/slapd.conf
...

#
# Indizes
#

index       objectClass                 eq
index       givenName,sn,cn,mail        eq,sub
```

Eine rudimentäre Verzeichnisdatenbank anlegen

Vor der eigentlichen Eingabe der Kontaktinformationen muss ein Wurzeleintrag erzeugt werden. OpenLDAP stellt grundsätzlich drei Mechanismen zur Manipulation der Verzeichnisdatenbank zur Verfügung:

- das LDAP-Protokoll selbst,
- einen Satz von Befehlen, die auf das LDAP-Protokoll aufsetzen (`ldapadd`, `ldapmodify`, `ldapdelete`, `ldapmodrdn`, `ldapcompare` und `ldapsearch`)
- und einen Satz von Befehlen, die direkt auf der Verzeichnisdatenbank arbeiten (`slapadd`, `slapcat` und `slapindex`)

Während die ersten beiden Mechanismen natürlich nur dann eingesetzt werden können, wenn der LDAP-Server `slapd` läuft, kann man die Befehle `slapadd`, `slapcat` und `slapindex` auch bei abgeschaltetem LDAP-Server verwenden.

Die drei Befehle lassen sich wie folgt einsetzen:

slapadd erzeugt neue Einträge in der Verzeichnisdatenbank. Eine Typüberprüfung findet nicht statt, so dass auch ungültige Einträge erzeugt werden können.

slapcat gibt den Inhalt der Datenbank aus.

slapindex erzeugt bzw. aktualisiert einen Index. Dieser Befehl muss verwendet werden, wenn ein Index zu einer bereits mit Daten gefüllten Verzeichnisdatenbank hinzugefügt worden ist.

Allen drei Befehlen ist gemein, dass man sicherstellen sollte, dass die Verzeichnisdatenbank während der Ausführung nicht parallel auch noch auf andere Weise verändert wird; am besten natürlich, indem man den LDAP-Server während Wartungsarbeiten mit diesen Befehlen abschaltet.

Um einen Wurzeleintrag zu erzeugen ist der `slapadd` demnach am besten geeignet. Ähnlich wie fast alle anderen in diesem Abschnitt genannten Befehle arbeitet dieser Befehl mit Eingabedaten im *LDAP Interchange Format* (kurz LDIF).

Bei LDIF handelt es sich um ein im RFC 2849 definiertes, textbasiertes Standardformat für den Austausch von Verzeichnisdaten. Eine LDIF-Datei besteht aus einer Auflistung von einzelnen Verzeichniseinträgen, die jeweils durch eine Leerzeile voneinander getrennt werden. Die Attribute eines Eintrags werden, beginnend mit dem DN, jeweils untereinander geschrieben. Der Attributname wird durch einen Doppelpunkt abgeschlossen, der dazugehörige Wert in die gleiche Zeile geschrieben wie der Attributname, wobei zwischen dem Doppelpunkt und dem Wert ein Leerzeichen vorgeschrieben ist.

Zusätzlich sind in LDIF-Dateien Kommentare (alle Zeilen, die mit einem `#` beginnen), sowie Steuerungsanweisungen für den LDIF-Parser erlaubt. Mit letzteren kann man u. a. angeben, ob ein neuer Eintrag hinzugefügt oder ein vorhandener Eintrag aktualisiert werden soll.

Der Wurzeleintrag für unser Adressbuch könnte in der LDIF-Notation z. B. so aussehen:

```
#
# Wurzeleintrag
#

dn: dc=example,dc=com
dc: example
objectClass: domain
```

Sofern man beabsichtigt, mittelfristig neben Kontaktinformationen auch andere Informationen in der Verzeichnisdatenbank abzulegen, sollte man ein zusätzliches Unterverzeichnis anlegen, in dem die Kontaktinformationen dann abgelegt werden können. Der Name des Unterverzeichnisses kann frei gewählt

werden, üblich ist dabei z. B. „people“. Die vollständige Datei mit der Definition des Wurzeleintrags und des Unterverzeichnisses hätte dann das folgende Aussehen:

```
$ cat root.ldif
#
# Wurzeleintrag
#

dn: dc=example,dc=com
dc: example
objectClass: domain

#
# Unterverzeichnis fuer Kontaktdaten
#
dn: ou=people,dc=example,dc=com
ou: people
objectClass: organizationalUnit
```

Die gewählten Klassen (`domain` für den Wurzeleintrag und `organizationalUnit`) spielen eine untergeordnete Rolle, da es sich bei diesen Einträgen lediglich um Organisationselemente ohne weitere Attribute handelt.

Um die Datenbank neu anzulegen muss man dann nur noch das entsprechende, mit Hilfe der `directory`-Anweisung in der Konfigurationsdatei spezifizierte Verzeichnis anlegen und den `slapadd` Befehl ausführen:

```
$ mkdir /Library/LDAPAdressbuch/Datenbank

$ sudo slapadd -v -l /Library/LDAPAdressbuch/root.ldif -f
                     /Library/LDAPAdressbuch/slapd.conf
added: "dc=kobylinski-consulting,dc=de" (00000001)
added: "ou=people,dc=kobylinski-consulting,dc=de" (00000002)
```

Die beim Aufruf von `slapadd` verwendeten Optionen haben dabei den folgenden Hintergrund:

-v (engl. *verbose*) veranlasst die Ausgabe etwas ausführlicherer Erfolgsmeldungen,

-l ... spezifiziert die Eingabedatei,

-f ... spezifiziert die Konfigurationsdatei. `slapadd` liest daraus u. a. den Pfad zur Verzeichnisdatenbank

Nachdem die Datenbank angelegt worden ist, kann man den LDAP-Server starten:

```
$ sudo /usr/libexec/slapd -d 256 -f
       /Library/LDAPAdressbuch/slapd.conf
@(#) $OpenLDAP: slapd 2.2.19 $
bdb_back_initialize: Sleepycat Software: Berkeley DB 4.2.52:
                                         (December  3, 2003)
bdb_db_init: Initializing BDB database
slapd starting
...
```

Die Option `-f` weist den LDAP-Server dabei an, anstelle der Standardkonfigurationsdatei `/etc/openldap/slapd.conf` die angegebene zu verwenden.

Um zu testen, ob der LDAP-Server wirklich funktioniert, kann man `ldapsearch` oder den LDAP Browser/Editor verwenden. Allzuviele Einträge darf man dabei natürlich nicht erwarten, sondern nur die, die man vorher mit Hilfe von `slapadd` hinzugefügt hat:

```
$ ldapsearch -x -h 127.0.0.1 -b "dc=example,dc=com"
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
dc: example
objectClass: domain

# people, example.com
dn: ou=people,dc=example,dc=com
ou: people
objectClass: organizationalUnit

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Dabei ist natürlich auffällig, dass die Ausgabe von `ldapsearch` genau dem LDIF-Format entspricht. Gefundene Einträge könnten auf diese Weise z. B. als Sicherungskopie gesichert werden. Will man das nicht, lässt sich das Aussehen der Ausgabe im gewissen Rahmen mit den Optionen `-L`, `-LL` und `-LLL` beeinflussen. Die erste Option beschränkt die Ausgabe auf eine einfachere Version von LDIF, die zweite blendet zusätzlich die Kommentarzeilen und die dritte auch die Versionsangabe aus. Um also nur die wesentlichsten Ergebnismerkmale auszugeben, müsste man den obigen Suchbefehl einfach um die Option `-LLL` ergänzen:

```
$ ldapsearch -LLL -x -h 127.0.0.1 -b "dc=example,dc=com"
dn: dc=example,dc=com
dc: example
objectClass: domain

dn: ou=people,dc=example,dc=com
ou: people
objectClass: organizationalUnit
```

Kontaktdaten eingeben

Sobald die Verzeichnisdatenbank in ihrer rudimentären Form funktioniert, kann man von `slapadd` auf `ldapadd` umsteigen, um die Daten nicht mehr direkt in die Datenbank, sondern über den LDAP-Server eintragen zu lassen.

Genau wie `slapadd` erwartet auch `ldapadd` eine Eingabedatei im LDIF-Format. Der für Kontaktinformationen übliche Datentyp bzw. die Klasse ist `inetOrgPerson`. Diese Klasse beinhaltet die wichtigsten Attribute, die vom Mac-OS-X-Adressbuch abgefragt werden. Insbesondere beinhaltet sie die Attribute `cn` (Vorname), `sn` (Nachname) und `mail`, deren Werte in der Ergebnisliste der Adressbuch-Anwendung dargestellt werden können.

Eine LDIF-Datei mit zwei Datensätzen hätte dann z. B. folgendes Aussehen:

```
$ cat /Library/LDAPAdressbuch/users.ldif
#
# Kontaktdaten von Rafael Kobylinski
#
dn: cn=Rafael Kobylinski,ou=people,dc=example,dc=com
cn: Rafael Kobylinski
sn: Kobylinski
givenName: Rafael
mail: kobylinski@example.com
objectClass: inetOrgPerson

#
# Kontaktdaten von Hans Mustermann
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
cn: Hans Mustermann
sn: Mustermann
givenName: Hans
mail: mustermann@example.com
objectClass: inetOrgPerson
```

Um diese zwei Datensätze in die Verzeichnisdatenbank zu integrieren muss man `ldapmodify` mit einigen Parametern aufrufen:

```
$ ldapadd -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f /Library/LDAPAdressbuch/users.ldif
adding new entry "cn=Rafael Kobylinski,ou=people,dc=example,
                  dc=com"

adding new entry "cn=Hans Mustermann,ou=people,dc=example,
                  dc=com"
```

Die Optionen `-x` und `-h 127.0.0.1` entsprechen in Syntax und Semantik genau denselben bereits bekannten Optionen von `ldapsearch`. Neu sind:

-D ... ist erforderlich, um Schreibzugriffe auf die Verzeichnisdatenbank zu authorisieren. Der angegebene DN entspricht dabei dem Parameter der Anweisung `rootdn` in der Konfigurationsdatei.

-w ... liefert das Passwort. Ein Hash des angegebenen Passworts steht als Parameter der Anweisung `rootpw` in der Konfigurationsdatei.

-f ... spezifiziert die Konfigurationsdatei.

Ein anschließender Aufruf von `ldapsearch` verifiziert, dass die Datensätze tatsächlich integriert worden sind:

```
$ ldapsearch -LLL -x -h 127.0.0.1 -b "dc=example,dc=com" \
> "(objectClass=inetOrgPerson)"
dn: cn=Rafael Kobylinski,ou=people,dc=example,dc=com
cn: Rafael Kobylinski
sn: Kobylinski
givenName: Rafael
mail: kobylinski@example.com
objectClass: inetOrgPerson

dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
cn: Hans Mustermann
sn: Mustermann
givenName: Hans
mail: mustermann@example.com
objectClass: inetOrgPerson
```

Ein entsprechendes Verbindungsprofil vorausgesetzt (Server 127.0.0.1 bzw. IP-Adresse des Server-Rechners, Suchbereich „dc=example,dc=com“ oder etwas präziser „ou=people,dc=example,dc=com“, vgl. Abb. 4.20) kann dann auch die Adressbuch-Anwendung für die Suche verwendet werden (Abb. 4.21).

LDAP-Server mit einem gültigen Verbindungsprofil werden von Apple Mail automatisch zur Ergänzung von unvollständig eingegebenen Email-Adressen herangezogen. Die entsprechenden Einstellungen können im Konfigurationsdialog „Verfassen“ von Apple Mail überprüft werden (Abb. 4.22).

Abb. 4.20. Das Verbindungsprofil für die eigene Verzeichnisdatenbank in den LDAP-Einstellungen der Adressbuch-Anwendung.

Abb. 4.21. Das Ergebnis einer LDAP-Suchanfrage in der frisch angelegten Verzeichnisdatenbank.

Abb. 4.22. LDAP-Server mit einem gültigen Verbindungsprofil werden von Apple Mail automatisch zur Ergänzung von unvollständig eingegebenen Email-Adressen herangezogen.

Kontaktdaten bearbeiten

Bereits vorhandene Datensätze lassen sich mit dem Befehl `ldapmodify` bearbeiten. Mit Hilfe von in LDIF-Dateien eingebetteten Steuerungsanweisungen lässt sich steuern, ob `ldapmodify` dabei Datensätze oder Attribute hinzufügen, löschen oder modifizieren soll.

`ldapmodify` liest eine LDIF-Datei mit Steuerungsanweisungen als Eingabe und entscheidet anhand der Steuerungsanweisung `changetype` für jeden in der Eingabe enthaltenen Datensatz einzeln, ob ein solcher Datensatz erzeugt, gelöscht, umbenannt (Änderung des RDN) oder modifiziert werden soll.

Die Steuerungsanweisung `changetype` wird dabei unmittelbar nach dem DN des Datensatzes wie ein Attribut geschrieben. Als Werte von `changetype` sind `add`, `delete`, `modify` und `modrdn` zugelassen.

Dabei bedeutet `changetype: add`, dass ein neuer Datensatz erzeugt werden soll. Entsprechend erwartet `ldapmodify` in den auf diese Steuerungsanweisung folgenden Zeilen die von `ldapadd` bekannte Liste von Attributen und Werten. Die auffällige Ähnlichkeit mit `ldapadd` ist dabei nicht zufällig. Tatsächlich sind `ldapadd` und `ldapmodify` exakt das gleiche Programm:

```
$ ls -l ‘which ldapadd‘ ‘which ldapmodify‘
-rwxr-xr-x   2 root  wheel  221400 Mar 21 08:01 /usr/bin/ldapadd
-rwxr-xr-x   2 root  wheel  221400 Mar 21 08:01 /usr/bin/ldapmodify
```

Sie unterscheiden sich lediglich dahingehend, dass `ldapadd` gleichbedeutend mit `ldapmodify -a` ist. Die Option `-a` weist dabei `ldapmodify` an, für jeden Datensatz implizit `changetype: add` anzunehmen. Ohne diese Option muss man diese Steuerungsanweisung explizit angeben.

Entsprechend hätte man unsere zwei Beispieldatensätze nach Ergänzung der Steuerungsanweisungen auch mit `ldapmodify` erzeugen können:

```
$ cat /Library/LDAPAdressbuch/mod_users.ldif
#
# Kontaktdaten von Rafael Kobylinski
#
dn: cn=Rafael Kobylinski,ou=people,dc=example,dc=com
changetype: add
cn: Rafael Kobylinski
sn: Kobylinski
givenName: Rafael
mail: kobylinski@example.com
objectClass: inetOrgPerson

#
# Kontaktdaten von Hans Mustermann
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: add
cn: Hans Mustermann
sn: Mustermann
givenName: Hans
mail: mustermann@example.com
objectClass: inetOrgPerson
```

```
$ ldapmodify -c -x -h 127.0.0.1 -D "cn=admin,ou=people,
                    dc=example,dc=com" -w admin -f
                    /Library/LDAPAdressbuch/mod_users.ldif
adding new entry "cn=Hans Mustermann,ou=people,dc=example,
                  dc=com"
```

Um einen Datensatz zu löschen, genügt die Angabe der DN und der Steuerungsanweisung `changetype: delete`. Um den Datensatz von „Hans Mustermann“ zu löschen, würde man also eine nur sehr kurze Eingabedatei benötigen[11]:

```
$ ldapsearch -LLL -x -h 127.0.0.1 -b "dc=example,dc=com"
                        "(cn=Hans*)"
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
cn: Hans Mustermann
sn: Mustermann
givenName: Hans
mail: mustermann@example.com
objectClass: inetOrgPerson

$ cat /Library/LDAPAdressbuch/del_hans.ldif
#
# Loeschen von Hans Mustermann
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: delete

$ ldapmodify -x -h 127.0.0.1 -D "cn=admin,ou=people,
                  dc=example,dc=com" -w admin -f
                  /Library/LDAPAdressbuch/del_hans.ldif
deleting entry "cn=Hans Mustermann,ou=people,dc=example,
                dc=com"

$ ldapsearch -LLL -x -h 127.0.0.1 -b "dc=example,dc=com"
                        "(cn=Hans*)"
$
```

Immer dann, wenn `ldapmodify` ohne die Option `-a` aufgerufen worden ist und keine `changetype`-Anweisung explizit angegeben wurde, wird `changetype: modify` verwendet. Diese Steuerungsanweisung setzt voraus, dass der mit Hilfe der DN spezifizierte Datensatz bereits existiert und dass Attribute dieses Datensatzes modifiziert werden sollen.

Um zu beschreiben, welche Attribute wie geändert werden sollen, muss die `changetype`-Anweisung mit einer weiteren Steuerungsanweisung vom Typ `add` (Hinzufügen), `replace` (Ändern) oder `delete` (Entfernen) konkretisiert werden.

Als Wert von `add`, `replace` und `delete` ist jeweils das zu modifizierende Attribut anzugeben. In der darauf folgenden Zeile muss dann auch beim Löschen das Attribut nochmals zusammen mit dem entsprechenden Wert angegeben werden. Da LDAP-Attribute auch mehrere Werte haben können,

[11] Zum Löschen kann alternativ auch der Spezialbefehl `ldapdelete` verwendet werden. `ldapdelete` unterstützt über die Option `-r` das automatische (rekursive) Löschen eines Unterverzeichnisses mitsamt Inhalt.

ist man auf diese Weise gezwungen, genau anzugeben, welcher der u. U. vielen Werte eines Attributs gelöscht werden soll.

Um „Hans Mustermann“ wieder schrittweise hinzuzufügen, könnten wir zunächst einen Basiseintrag erzeugen und diesen Eintrag dann um weitere Attribute ergänzen. Als erstes spezifizieren wir den Basiseintrag in der Datei `add_hans.ldif` und erzeugen diesen Eintrag mit `ldapmodify`:

```
$ cat /Library/LDAPAdressbuch/add_hans.ldif
#
# Hinzufuegen
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: add
cn: Hans Mustermann
sn: Mustermann
objectClass: inetOrgPerson

$ ldapmodify -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f /Library/LDAPAdressbuch/add_hans.ldif
adding new entry "cn=Hans Mustermann,ou=people,dc=example,
                  dc=com"
```

Anschließend spezifizieren wir in der Datei `mod_hans.ldif` ein zusätzliches Attribut `mail` und verwenden `ldapmodify`, um es hinzuzufügen:

```
$ cat /Library/LDAPAdressbuch/mod_hans.ldif
#
# Ergänzen
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: modify
add: mail
mail: mustermann@example.com

$ ldapmodify -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f /Library/LDAPAdressbuch/mod_hans.ldif
modifying entry "cn=Hans Mustermann,ou=people,dc=example,
                 dc=com"
```

Und schließlich ändern wir mit Hilfe der Datei `modrep_hans.ldif` den Wert des Attributs `mail`[12]:

```
$ cat /Library/LDAPAdressbuch/modrep_hans.ldif
#
# Aendern
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: modify
replace: mail
mail: hans.mustermann@example.com
```

[12] Bei der Replace-Operation werden alle Werte des angegebenen Attributs gelöscht und durch den neuen Wert oder die neuen Werte ersetzt.

```
$ ldapmodify -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f /Library/LDAPAdressbuch/modrep_hans.ldif
modifying entry "cn=Hans Mustermann,ou=people,dc=example,
                 dc=com"
```

Das Ergebnis:

```
$ ldapsearch -LLL -x -h 127.0.0.1 -b "dc=example,dc=com"
                        "(cn=Hans*)"
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
cn: Hans Mustermann
sn: Mustermann
objectClass: inetOrgPerson
mail: hans.mustermann@example.com
```

Eine `ldapmodify`-Eingabedatei wird einfach von oben nach unten abgearbeitet. Eine solche Eingabedatei kann deshalb nicht nur Änderungsanweisungen für mehrere Datensätze, sondern auch mehrere unterschiedliche Änderungsanweisungen für ein und denselben Datensatz beinhalten. Es ist sogar möglich, innerhalb einer einzigen Eingabedatei einen Datensatz erst zu erzeugen, um ihn dann in einem weiter unten liegenden Abschnitt der Datei zu modifizieren.

Mehrere, den gleichen Datensatz betreffende Modifikationen vom Typ `modify` können zu einer einzigen Transaktion zusammengefasst werden. Die Transaktion wird dabei als Ganzes vorgenommen, so dass inkonsistente Zwischenzustände vermieden werden können. Die zusammengehörigen Operationen werden dazu nach der einleitenden `changetype: modify`-Anweisung direkt untereinander geschrieben und durch Zeilen mit einem einzelnen `-` voneinander getrennt.

Man kann also z. B. ohne weiteres in einem Schritt Attribute entfernen und hinzufügen:

```
$ cat /Library/LDAPAdressbuch/modmult_hans.ldif
#
# Mehrfach modifizieren
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: modify
delete: mail
mail: hans.mustermann@example.com
-
add: mail
mail: mustermann@example.com
-
add: telephoneNumber
telephoneNumber: (089) 123456-78
-
add: jpegPhoto
jpegPhoto:< file:///Users/hans/Pictures/Hans.jpg

$ ldapmodify -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f modmult_hans.ldif
```

```
modifying entry "cn=Hans Mustermann,ou=people,dc=example,
                 dc=com"
```

Im obigen Beispiel wird die alte Email-Adresse gelöscht und durch eine neue ersetzt (diesmal mit einem expliziten `delete` und `add` und nicht mit einem `replace`). Zusätzlich werden eine Telefonnummer und ein Passbild – mit Binärdaten aus der lokalen Datei /Users/hans/Pictures/Hans.jpg – hinterlegt[13].

Auf diese Weise lassen sich alle Attribute bis auf `cn` ändern. Um dieses Attribut zu modifizieren, muss man die spezielle Steuerungsanweisung `changetype: modrdn` verwenden, da `cn` als RDN (d. h. als eindeutiges Identifikationsmerkmal des Eintrags) verwendet wird. Anschließend spezifiziert man mit `newrdn` den neuen RDN und entscheidet mit `deleteoldrdn`, ob der bisher als RDN verwendete Wert gelöscht werden soll oder nicht:

```
$ cat /Library/LDAPAdressbuch/modrdn_hans.ldif
#
# RDN Aendern
#
dn: cn=Hans Mustermann,ou=people,dc=example,dc=com
changetype: modrdn
newrdn: cn=Johannes Mustermann
deleteoldrdn: 1
$ ldapmodify -x -h 127.0.0.1 \
> -D "cn=admin,ou=people,dc=example,dc=com" \
> -w admin \
> -f modrdn_hans.ldif
modifying rdn of entry "cn=Hans Mustermann,ou=people, dc=example,dc=com"
modrdn completed
```

Zusammenfassend lässt sich sagen, dass man mit `ldapmodify` jedwede, auch sehr umfangreiche, Änderung am Datenbestand durchführen kann. Insbesondere beim Einsatz von Skripten lassen sich Änderungen, die sehr viele Datensätze betreffen, schnell und effizient durchführen.

Soll nur ein einzelnes Attribut geändert werden, kann man aber auch auf den bereits erwähnten LDAP Browser/Editor zurückgreifen. Mit Hilfe dieses Werkzeuges lassen sich die Verzeichnisdaten nicht nur betrachten, sondern, eine passende Authentifizierung vorausgesetzt, auch ändern.

Um Änderungen durchführen zu können, muss man im Verbindungsprofil den „Anonymous Bind“ abschalten und anschließend unter „User Info“ die Werte von `rootdn` und `rootpw` eintragen. Sofern die Option „append base DN“ aktiv ist, genügt es, „cn=admin,ou=people“ einzutragen (Abb. 4.23).

Mit dem geänderten Verbindungsprofil baut man dann die Verbindung wie gewohnt auf und kann durch einen Doppelklick auf ein Attribut einen Änderungsdialog aufrufen. Im Änderungsdialog kann man dann den Attributwert bequem editieren (Abb. 4.24).

[13] Die Adressbuch-Anwendung zeigt das Passbild nur in der Detailansicht der Visitenkarte. Diese Detailansicht lässt sich durch einen Doppelklick auf ein LDAP-Suchergebnis oder nach dem Import der Visitenkarte in das lokale Adressbuch anzeigen.

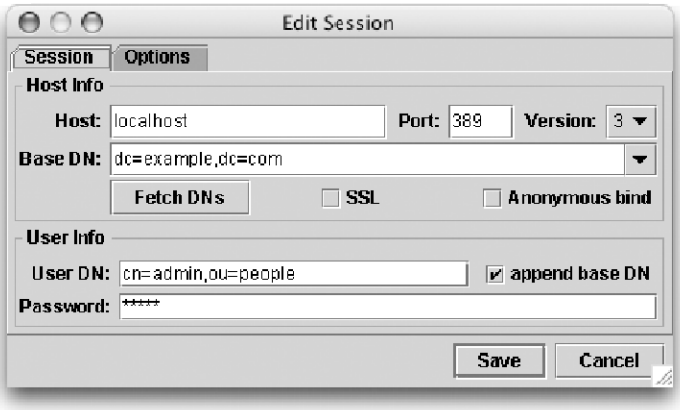


Abb. 4.23. Ein authentifiziertes Verbindungsprofil ist erforderlich, um Verzeichnisdaten zu ändern.

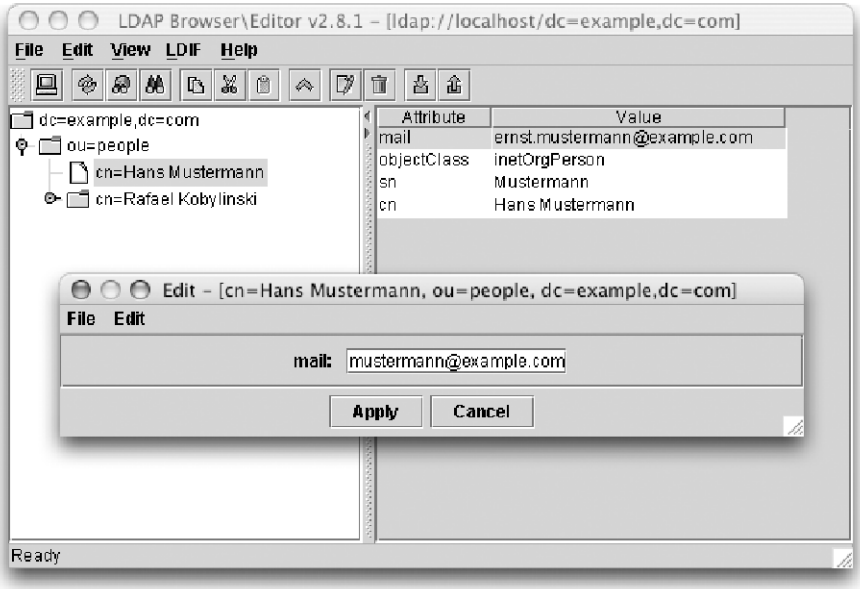


Abb. 4.24. Im Änderungsdialog kann man dann den Attributwert bequem editieren.

4.2.7 Netzwerkbenutzer mit Netzwerkverzeichnissen anlegen

Einträge für Netzwerkbenutzer lassen sich in einer über LDAP zugänglichen Verzeichnisdatenbank ebenso einfach ablegen wie in einer NetInfo-Datenbank. Grundsätzlich kann man einen beliebigen, LDAP-konformen Ver-

zeichnisdienst nutzen oder mit Hilfe des mitgelieferten OpenLDAP-Pakets eine neue Verzeichnisdatenbank von Grund auf neu erstellen.

Um eine auf OpenLDAP basierende Verzeichnisdatenbank von Grund auf neu zu erstellen, ist es sinnvoll, zunächst einen Ablageort für die Konfigurations- und Datenbankdateien zu definieren, z. B.:

```
$ mkdir -p /Library/LDAPVerzeichsdienst/Datenbank
```

In dem neu angelegten Verzeichnis kann man dann die Konfigurationsdatei für den LDAP-Server ablegen, `slapd.conf`.

In dieser Datei müssen die verwendeten Klassen deklariert, die Protokollierungsparameter, die Datenbankparameter, die Indexparameter und gegebenenfalls weitere Parameter definiert werden.

Die notwendigen Klassendeklarationen entsprechen den Klassendeklarationen in der Standardkonfigurationsdatei `/etc/openldap/slapd.conf`, nämlich:

```
#
# Klassen und Attribute
#

include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/samba.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/apple.schema
```

Die Protokollierungsparameter stellen sicher, dass die Prozessnummer und die beim Aufruf verwendeten Argumente im richtigen Verzeichnis abgelegt werden:

```
#
# Protokollierung
#

pidfile         /Library/LDAPVerzeichnisdienst/slapd.pid
argsfile        /Library/LDAPVerzeichnisdienst/slapd.args
```

Die Datenbankparameter legen die Art und den Ablageort der Datenbank fest. Das Suffix ordnet die Datenbank in ein globales Namensschema ein – wie üblich kann man hier den eigenen DNS-Namen verwenden. Darüber hinaus wird ein Benutzername (`root`) sowie das dazugehörige Passwort (hier im Beispiel auch `root`[14]) definiert, mit dem Schreibzugriffe erfolgen dürfen:

```
#
# Datenbank
#

database        bdb

suffix          "dc=example,dc=com"

rootdn          "cn=root,ou=users,dc=example,dc=com"
```

[14] Der Passwort-Hash kann mit dem Befehl *slappasswd* erstellt werden.

```
rootpw          {SSHA}hT1IhBT1k4muiSJn9mTBCrn6MRZvcS1h

directory       /Library/LDAPVerzeichnisdienst/Datenbank
```

Die Indizes stellen sicher, dass auf die indizierten Attribute bezogene Suchanfragen wirklich schnell beantwortet werden können. Vor dem Hintergrund, dass auf einem an einen Verzeichnisdienst angebundenen Client selbst einfache Befehle wie z. B. `ls -l` Suchanfragen generieren, ist die Bedeutung der Verarbeitungsgeschwindigkeit an dieser Stelle nicht zu unterschätzen. Die folgende Anweisung definiert Indizes für einige besonders häufig gesuchte Attribute:

```
#
# Indizes
#

index           objectClass,cn,uid,uidNumber,gidNumber    eq
```

Entsprechend sieht eine vollständige OpenLDAP-Konfigurationsdatei für einen einfachen, LDAP-basierten Verzeichnisdienst wie folgt aus:

```
$ cat /Library/LDAPVerzeichnisdienst/slapd.conf
#
# slapd.conf für einen zentralen Verzeichnisdienst
#

#
# Klassen und Attribute
#

include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/samba.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/apple.schema

#
# Protokollierung
#

pidfile         /Library/LDAPVerzeichnisdienst/slapd.pid
argsfile        /Library/LDAPVerzeichnisdienst/slapd.args

#
# Datenbank
#

database        bdb

suffix          "dc=example,dc=com"

rootdn          "cn=root,ou=users,dc=example,dc=com"
rootpw          {SSHA}hT1IhBT1k4muiSJn9mTBCrn6MRZvcS1h

directory       /Library/LDAPVerzeichnisdienst/Datenbank
```

```
#
# Indizes
#

index           objectClass,cn,uid,uidNumber,gidNumber    eq
```

Um die Verzeichnisdatenbank tatsächlich zu erstellen, kann man mit Hilfe des Befehls `slapadd` einige rudimentäre Einträge erzeugen. In unserem Fall ist es sinnvoll, vor dem Start des LDAP-Servers auf diese Weise neben dem obligatorischen Wurzel-Element zumindest zwei Unterelemente als Verzeichnisse für Benutzer- sowie für Automount-Einträge einzurichten:

```
$ cat /Library/LDAPVerzeichnisdienst/root.ldif
#
# Wurzeleintrag
#
dn: dc=example,dc=com
dc: example
objectClass: domain

#
# Unterverzeichnis fuer Benutzer
#
dn: cn=users,dc=example,dc=com
cn: users
objectClass: container

#
# Unterverzeichnis fuer Automounts
#
dn: cn=mounts,dc=example,dc=com
cn: mounts
objectClass: container
```

Die Datenbank kann dann mit der folgenden Anweisung erstellt werden:

```
$ sudo slapadd -v \
> -l /Library/LDAPVerzeichnisdienst/root.ldif \
> -f /Library/LDAPVerzeichnisdienst/slapd.conf
added: "dc=example,dc=com" (00000001)
added: "cn=users,dc=example,dc=com" (00000002)
added: "cn=mounts,dc=example,dc=com" (00000003)
```

Um zu verifizieren, dass die Datenbank tatsächlich erstellt worden ist, kann man in einem separaten Terminalfenster den LDAP-Server starten. Die Option `-d 256` belässt den LDAP-Server im Vordergrund und protokolliert alle bearbeiteten Suchanfragen:

```
$ sudo /usr/libexec/slapd -d 256 -f
       /Library/LDAPVerzeichnisdienst/slapd.conf
@(#) $OpenLDAP: slapd 2.2.19 $
bdb_back_initialize: Sleepycat Software: Berkeley DB 4.2.52:
                                         (December  3, 2003)
bdb_db_init: Initializing BDB database
slapd starting
...
```

Sofern alles geklappt hat, lässt sich die LDAP-Datenbank im ursprünglichen Terminalfenster mit Hilfe der Anweisung `ldapsearch` durchsuchen:

```
$ ldapsearch -x -LLL -h 127.0.0.1 -b "dc=example,dc=com"
dn: dc=example,dc=com
dc: example
objectClass: domain

dn: cn=users,dc=example,dc=com
cn: users
objectClass: container

dn: cn=mounts,dc=example,dc=com
cn: mounts
objectClass: container
```

Das Ergebnis zeigt, dass die weiter oben angelegten drei Objekte tatsächlich gefunden worden sind. Sofern der LDAP-Server mit der Option `-d 256` gestartet worden ist, kann man serverseitig eine den Parametern von `ldapsearch` entsprechende Ausgabe wie nachfolgend dargestellt beobachten:

```
...
conn=0 fd=13 ACCEPT from IP=127.0.0.1:49230 (IP=0.0.0.0:389)
conn=0 op=0 BIND dn="" method=128
conn=0 op=0 RESULT tag=97 err=0 text=
conn=0 op=1 SRCH base="dc=example,dc=com" scope=2 deref=0
            filter="(objectClass=*)"
conn=0 op=1 SEARCH RESULT tag=101 err=0 nentries=3 text=
conn=0 op=2 UNBIND
conn=0 fd=13 closed
...
```

Anders als bei der Kontaktdatenbank erzeugen wir die Datenbankeinträge nachfolgend mit `slapadd`. Da `slapadd` den LDAP-Server umgeht und direkt in die Datenbank schreibt, ist es an dieser Stelle sinnvoll, den LDAP-Server mittels Control-C zu beenden.

```
^Cslapd shutdown: waiting for 0 threads to terminate
slapd stopped.
```

Damit etwaige Netzwerkbenutzer auf Ihre zentralen Benutzerverzeichnisse zugreifen können, muss die Verzeichnisdatenbank einen entsprechenden Automount-Eintrag enthalten. Dazu muss im für diese Art von Einträgen vorgesehenen Unterverzeichnis `mounts` ein Eintrag der Klasse `mount` mit einer Reihe von Attributen erstellt werden, der den Speicherort der Freigabe sowie die erforderlichen Automount-Optionen beinhaltet.

```
$ cat /Library/LDAPVerzeichnisdienst/mounts.ldif
#
# Automount fuer zentrale Benutzerverzeichnisse
#
dn: cn=Tiger-Server.local:/Shares/Users,cn=mounts,dc=example,dc=com
cn: Tiger-Server.local:/Shares/Users
mountDirectory: /Network/Servers
mountType: url
```

```
mountOption: net
mountOption: url==afp://;AUTH=NO%20USER%20AUTHENT@Tiger-Server.
             local/Users
objectClass: mount
```

Die innerhalb der Automount-Einträge verwendeten LDAP-Attribute entsprechen eins zu eins den bereits im Abschn. 4.1.7 diskutierten NetInfo-Attributen. Die einander entsprechenden Attributpaare lassen sich am einfachsten der von der NetInfo-Bridge verwendeten Schema-Definition entnehmen:

```
$ cat /etc/openldap/schema/netinfo.schema
...
#
# Schema mapping for NetInfo mounts.
#
objectclassmap /mounts mount
attributemap /mounts dir mountDirectory
attributemap /mounts vfstype mountType
attributemap /mounts opts mountOption
attributemap /mounts dump_freq mountDumpFrequency
attributemap /mounts passno mountPassNo
...
```

Um einen Automount-Eintrag in der Verzeichnisdatenbank zu erstellen, kann die folgende Anweisung verwendet werden:

```
$ sudo slapadd -v \
> -l /Library/LDAPVerzeichnisdienst/mounts.ldif \
> -f /Library/LDAPVerzeichnisdienst/slapd.conf
added: "cn=Tiger-Server.local:/Shares/Users,cn=mounts,
        dc=example,dc=com" (00000004)
```

Um das Ergebnis dieser Anweisung zu verifizieren, kann man wie oben den LDAP-Server in einem eigenen Terminal-Fenster starten, und eine Suchanfrage starten.

```
$ sudo /usr/libexec/slapd -d 256 -f
       /Library/LDAPVerzeichnisdienst/slapd.conf
@(#) $OpenLDAP: slapd 2.2.19 $
bdb_back_initialize: Sleepycat Software: Berkeley DB 4.2.52:
                                         (December  3, 2003)
bdb_db_init: Initializing BDB database
slapd starting

...
```

Die Suchanfrage kann in diesem Fall auf Objekte vom Typ `mount` eingeschränkt werden (Parameter `"(objectClass=mount)"`). Gleichzeitig kann man als Startpfad gleich das Unterverzeichnis `mounts` (Parameter `-b ...`) anstelle der Wurzel der Verzeichnisdatenbank angeben, um den Suchvorgang zu beschleunigen[15]:

[15] Natürlich spielt diese Optimierung bei der Beispieldatenbank, die nur einige wenige Objekte enthält, überhaupt keine Rolle. Anders sieht das allerdings im Pra-

```
$ ldapsearch -x -LLL -h 127.0.0.1 -b "cn=mounts,dc=example,dc=com"
                        "(objectClass=mount)"
dn: cn=Tiger-Server.local:/Shares/Users,cn=mounts,dc=example,dc=com
cn: Tiger-Server.local:/Shares/Users
mountDirectory: /Network/Servers
mountType: url
mountOption: net
mountOption: url==afp://;AUTH=NO%20USER%20AUTHENT@Tiger-Server.
             local/Users
objectClass: mount
```

Im letzten Schritt muss schließlich ein Eintrag (bzw. in der Praxis normalerweise mehrere Einträge) für einen Netzwerkbenutzer angelegt werden.

Die folgende LDIF-Datei enthält die für einen einfachen Netzwerkbenutzer relevanten Attribute:

```
$ cat /Library/LDAPVerzeichnisdienst/users.ldif
#
# Benutzerdaten fuer Test Benutzer
#
dn: cn=Test Benutzer,cn=users,dc=example,dc=com
cn: Test Benutzer
sn: Test
uid: test
uidNumber: 2048
gidNumber: 20
userPassword: EeEzXNCM.8x1Q
apple-user-authenticationhint: test
homeDirectory: /Network/Servers/Tiger-Server.local/Shares/
               Users/test
apple-user-homeurl:  <homeDir><url>afp://Tiger-Server.local/
                     Users</url><path>test</path></homeDir>
loginShell: /bin/bash
objectClass: apple-user
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: extensibleObject
```

Das Attribut `userPassword` enthält dabei das Benutzerpasswort (hier `test`) im Crypt-Format:

```
$ openssl passwd -salt Ee
Password:
EeEzXNCM.8x1Q
```

Auch hier lohnt sich ein Blick in die von der NetInfo-Bridge verwendeten Schema-Definitionen, um auf die Schnelle die für die Definition des Eintrags notwendigen Werte für das `objectClass`-Attribut sowie die korrekten Namen für die übrigen Attribute zu verifizieren.

```
$ cat /etc/openldap/schema/netinfo.schema
...
```

xisbetrieb mit großen Verzeichnisdatenbanken aus, die Tausende von Objekten beinhalten.

```
#
# Schema mapping for NetInfo users.
#
objectclassmap /users inetOrgPerson posixAccount
               shadowAccount apple-user extens
ibleObject
attributemap /users name uid
attributemap /users realname cn
attributemap /users uid uidNumber
attributemap /users gid gidNumber
attributemap /users home homeDirectory
attributemap /users shell loginShell
...
```

Genauso wie die Werte der oben vorgestellten Attribute von LDAP-Automount-Einträgen, entsprechen auch die Werte der Attribute der LDAP-Benutzer-Einträge eins zu eins den entsprechenden NetInfo-Attributen (siehe Abschn. 4.1.5).

Ein Benutzer-Eintrag wird dann natürlich genauso wie ein Automount-Eintrag erstellt:

```
$ sudo slapadd -v \
> -l /Library/LDAPVerzeichnisdienst/users.ldif  \
> -f /Library/LDAPVerzeichnisdienst/slapd.conf
added: "cn=Test Benutzer,cn=users,dc=example,dc=com"
       (00000005)
```

Um zu verifizieren, dass in der Datenbank tatsächlich ein Benutzer-Eintrag angelegt wurde, kann man wie oben `ldapsearch` verwenden:

```
$ ldapsearch -x -LLL -h 127.0.0.1 -b "cn=users,dc=example,dc=com"
                        "(objectClass=apple-user)"
dn: cn=Test Benutzer,cn=users,dc=example,dc=com
cn: Test Benutzer
sn: Test
uid: test
uidNumber: 2048
gidNumber: 20
userPassword:: RWVFelhOQ0OuOHgxUQ==
apple-user-authenticationhint: test
homeDirectory: /Network/Servers/Tiger-Server.local/Shares/
               Users/test
apple-user-homeurl:: PGhvbWVEaXI+
 PHVybD5hZnA6Ly9UaWdlci1TZXJ2ZXIubG9jYWwvVXNlc
 nM8L3VybD48cGF0aD5OZXN0PC9wYXRoPjwvaG9tZURpcj4=
loginShell: /bin/bash
objectClass: apple-user
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: extensibleObject
```

Die Werte der beiden Attribute `userPassword` und `apple-user-homeurl` werden im Base64-Format angezeigt (diese Kodierung wird durch einen zusätzlichen Doppelpunkt hinter dem Attributnamen gekennzeichnet). Base64 ist ein Verfahren zur eindeutigen Abbildung von 8-Bit-Binärdaten auf 7-Bit-ASCII-Zeichen und wird heutzutage z. B. für die Kodierung von Email-Anhängen verwendet.

Mit Base64 kodierte Daten lassen sich mit Hilfe der Anweisung `openssl base64 -d` grundsätzlich sehr einfach dekodieren[16]:

```
$ echo "RWVFelhOQ0OuOHgxUQ==" | openssl base64 -d; echo
EeEzXNCM.8x1Q
```

Bei der Kodierung wird nach jeweils 76 Zeichen ein Zeilenumbruch eingefügt. Längere Werte wie der Wert des Attributs `apple-user-homeurl` sind daher etwas aufwändiger zu dekodieren:

```
$ ldapsearch  -x -LLL -h 127.0.0.1 -b "dc=example,dc=com" \
> "(&(objectClass=apple-user)(uid=test))" apple-user-homeurl \
> | sed 's/apple-user-homeurl::\//' \
> | grep -v 'dn:' | openssl base64 -d; echo
<homeDir><url>afp://Tiger-Server.local/Users</url><path>test</path>
</homeDir>
```

In diesen Fällen kann man mit Hilfe der Option `-t` den Wert des binären Attributs direkt in einer temporären Datei ablegen lassen:

```
$ ldapsearch -t -x -LLL -h 127.0.0.1 \
> -b "dc=example,dc=com" \
> "(&(objectClass=apple-user)(uid=test))" \
> apple-user-homeurl
dn: cn=Test Benutzer,cn=users,dc=example,dc=com
apple-user-homeurl:<
  file:///var/tmp//ldapsearch-apple-user-homeurl-Y5oEBZ
```

Die von `ldapsearch` erstellte temporäre Datei enthält dann die unkodierten Rohdaten, die direkt, ohne zusätzliche Dekodierungsschritte, weiterverarbeitet werden können:

```
$ cat /var/tmp//ldapsearch-apple-user-homeurl-Y5oEBZ; echo
<homeDir><url>afp://Tiger-Server.local/Users</url>
<path>test</path></homeDir>
```

4.2.8 Zentrale Verzeichnisdatenbank verwenden

Um eine zentrale Verzeichnisdatenbank für die Anmeldung auf einer Client-Machine verwenden zu können, muss man den Client mit Hilfe des Dienstprogramms Verzeichnisdienste entsprechend konfigurieren.

Bei der Verwendung eines LDAP-basierten Verzeichnisdienstes muss man zuerst die Verbindungsparameter zum Verzeichnisdienst angeben und dann den Authentifizierungspfad des Systems um den neuen Verzeichnisdienst ergänzen.

Zusätzlich ist es erforderlich, die Abbildung von abstrakten Open-Directory-Datentypen auf LDAP-Klassen sowie die Abbildung von Open-Directory-Attributen auf LDAP-Attribute anzugeben. Auf diese Weise lassen sich auf verhältnismäßig einfache Weise bestehende Verzeichnisdienste verwenden, ohne Veränderungen an diesen Verzeichnisdiensten durchführen zu müssen.

[16] Die abschließende echo-Anweisung erzeugt hier nur einen zusätzlichen Zeilenumbruch und kann natürlich weggelassen werden.

Den Dialog zur Konfiguration der LDAP-Verbindungsparameter erreicht man im Dienstprogramm Verzeichnisdienste durch einen Doppelklick auf den Eintrag „LDAPv3“ im Bereich „Dienste“ (Abb. 4.25)[17]. Dabei sollte man sicherstellen, dass nach erfolgter Konfiguration die Checkbox neben diesem Eintrag auch aktiviert ist und Open Directory tatsächlich Daten über die konfigurierte LDAP-Verbindung bezieht.

Im Konfigurationsdialog des LDAP-Plugins kann man die LDAP-Verbindungsparameter entweder automatisch über DHCP beziehen, oder manuell ein LDAP-Verbindungsprofil erstellen.

Jedes Verbindungsprofil muss über einen Konfigurationsnamen, einen Servernamen sowie über eine Vorschrift zur Abbildung der abstrakten Open-Directory-Klassen und -Attribute auf die vom jeweiligen Verzeichnisdienst tatsächlich verwendeten Klassen und Attribute verfügen. Zusätzlich kann über eine Checkbox die SSL-Verschlüsselung aktiviert werden (Abb. 4.26). In Mac OS X 10.4 kann man im Erstellungsdialog direkt angeben, ob das Profil für die Authentifizierung und/oder für die Suche nach Kontaktinformationen verwendet werden soll.

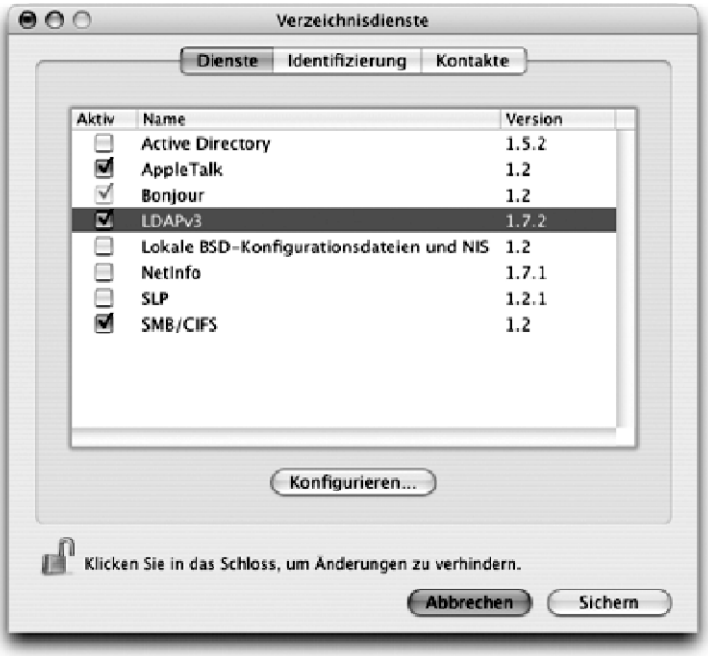


Abb. 4.25. Den Dialog zur Konfiguration der LDAP-Verbindungsparameter erreicht man im Dienstprogramm Verzeichnisdienste durch einen Doppelklick auf den Eintrag „LDAPv3“ im Bereich „Dienste“.

[17] Der Button „Konfigurieren ...“ öffnet das gleiche Konfigurationsfenster.

Abb. 4.26. Jedes Verbindungsprofil muss über einen Konfigurationsnamen, einen Servernamen sowie über eine Vorschrift zur Abbildung der abstrakten Open-Directory-Klassen und -Attribute auf die vom jeweiligen Verzeichnisdienst tatsächlich verwendeten Klassen und Attribute verfügen. Zusätzlich kann über eine Checkbox die SSL-Verschlüsselung aktiviert werden.

Der Konfigurationsname kann frei gewählt werden. Der Servername kann als DNS-Hostname oder als IP-Adresse angegeben werden. Als Abbildungsvorschriften stehen die Optionen „Vom Server“, „Active Directory“, „Open Directory Server“, „RFC 2307 (Unix)“ sowie „Eigene“ zur Verfügung.

Jede dieser Abbildungsvorschriften legt fest, auf welche Art und Weise die abstrakten Open-Directory-Klassen und -Attribute auf die vom jeweiligen LDAP-Server tatsächlich verwendeten Klassen und Attribute abzubilden sind.

Intern verwendet Mac OS X, und zwar unabhängig davon in welchem Verzeichnisdienst die Verzeichnisdaten tatsächlich gespeichert sind, ein festes Schema von Klassen und Attributen, um auf die Verzeichnisdaten zuzugreifen.

Dieses feste Schema erlaubt den unter Mac OS X ablaufenden Anwendungen auf beliebige Verzeichnisdienste zuzugreifen, ohne dass die Anwendungen Kenntnisse über die interne Struktur der Daten eines jeden Verzeichnisdienstes haben müssten.

Wird eine der Optionen „Active Directory“, „Open Directory Server“ oder „RFC 2307 (Unix)“ als Abbildungsvorschrift gewählt, verwendet Mac OS X eine feste Abbildung, um das interne Open-Directory-Schema auf das Sche-

ma des verwendeten LDAP-Servers abzubilden. Dabei ist die Abbildungsvorschrift „Active Directory“ mit Microsoft-Active-Directory-Servern[18] und die Abbildungsvorschrift „RFC 2307 (Unix)“ mit vielen Unix-basierten LDAP-Servern kompatibel. Bei der Abbildungsvorschrift „Open Directory Server“ handelt es sich um die Apple-spezifische Standardabbildungsvorschrift, die vom in Mac-OS-X-Server integrierten Open-Directory-Server verwendet wird.

Die Option „Vom Server“ definiert keine feste Abbildungsvorschrift, sondern weist den Client an, die Abbildungsdefinition direkt vom Server zu laden. Die Option „Eigene“ erlaubt schließlich die Definition einer eigenen Abbildungsvorschrift (Abb. 4.27).

Die in diesem Kapitel definierten Automount- und Benutzer-Einträge sind mit der Standardabbildungsvorschrift kompatibel. Man kann daher einfach die Option „Open Directory Server“ auswählen, um die Verzeichnisinformationen beziehen zu können. Als *Suchbeginn-Suffix* ist dabei die DN des Wurzelverzeichnisses anzugeben, also z. B. `dc=example,dc=com`.

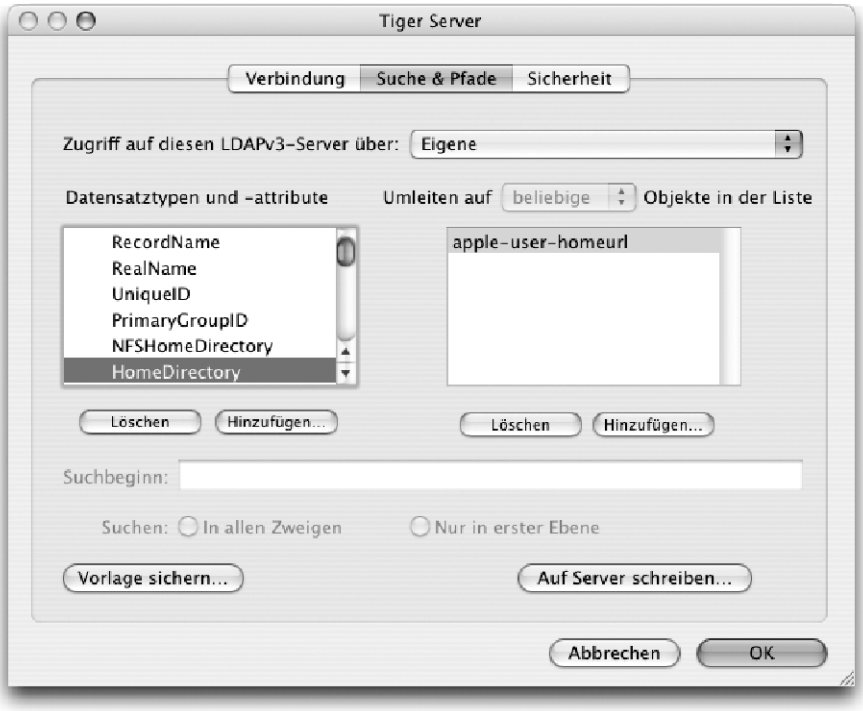


Abb. 4.27. Die Option „Eigene“ erlaubt die Definition einer eigenen Abbildungsvorschrift.

[18] Über das LDAP-Plugin ist eine einfache Integration mit Active-Directory-Servern möglich. Für eine umfassende Integration mit Active-Directory-basierenden IT-Infrastrukturen sollte man das dedizierte Active-Directory-Plugin verwenden.

Damit das LDAP-Verbindungsprofil für die Authentifizierung verwendet wird, muss man es zum Authentifizierungspfad hinzufügen. Dazu wählt man im Hauptfenster des Dienstprogramms Verzeichnisdienste den Bereich „Identifizierung“ und überprüft, ob ein Eintrag für das Verbindungsprofil bereits existiert. Wenn das nicht der Fall ist, kann man auf den Button „Hinzufügen...“ klicken. Im daraufhin erscheinenden Dialog kann man dann den LDAP-Verzeichnisdienst aus der Liste auswählen, z. B. als `/LDAPv3/Tiger-Server.local`.

Um die Konfiguration zu überprüfen, kann man den Befehl `dscl` im interaktiven Modus verwenden. Ist die Verbindung hergestellt, kann man mit den Befehlen `cd`, `ls` und `read` auf einfache Weise in der Datenbank des LDAP-Servers navigieren:

```
$ dscl localhost
 > cd LDAPv3/Tiger-Server.local/
/LDAPv3/Tiger-Server.local > read Users/test/
apple-user-authenticationhint: test
apple-user-homeurl: <homeDir><url>afp://Tiger-Server.local/
                    Users</url><path>test</path></homeDir>
cn: Test Benutzer
gidNumber: 20
homeDirectory: /Network/Servers/Tiger-Server.local/Shares/
               Users/test
loginShell: /bin/bash
objectClass: apple-user inetOrgPerson posixAccount shadowAccount
             extensibleObject
sn: Test
uid: test
uidNumber: 2048
userPassword: EeEzXNCM.8x1Q
AppleMetaNodeLocation: /LDAPv3/Tiger-Server.local
AuthenticationHint: test
HomeDirectory: <homeDir><url>afp://Tiger-Server.local/Users</url>
               <path>test</path></homeDir>
LastName: Test
NFSHomeDirectory: /Network/Servers/Tiger-Server.local/Shares/
                  Users/test
Password: EeEzXNCM.8x1Q
PrimaryGroupID: 20
RealName: Test Benutzer
RecordName: test Test Benutzer
RecordType: dsRecTypeStandard:Users
UniqueID: 2048
UserShell: /bin/bash
/LDAPv3/Tiger-Server.local > read Mounts/Tiger-Server.local:
                             \/Shares\/Users/
cn: Tiger-Server.local:/Shares/Users
mountDirectory: /Network/Servers
mountOption: net url==afp://;AUTH=NO%20USER%20AUTHENT@Tiger-Server.
             local/Users
mountType: url
objectClass: mount
AppleMetaNodeLocation: /LDAPv3/Tiger-Server.local
PasswordPlus: ********
RecordName: Tiger-Server.local:/Shares/Users
RecordType: dsRecTypeStandard:Mounts
```

```
VFSLinkDir: /Network/Servers
VFSOpts: net url==afp://;AUTH=NO%20USER%20AUTHENT@Tiger-Server.
         local/Users
VFSType: url
/LDAPv3/Tiger-Server.local > quit
Goodbye
```

Nachdem man verifiziert hat, dass die Verbindung zum LDAP-Server in Ordnung ist, kann man mit `dscl` überprüfen, ob die im LDAP-Verzeichnis eingetragenen Benutzer tatsächlich für die Authentifizierung verwendet werden können. Man muss dazu das Pseudoverzeichnis `Search` durchsuchen, in dem man die Summe der Einträge aller durchsuchten Verzeichnisse findet. Im Verzeichnis `/Search/Users/` sollte man demnach neben den lokalen Benutzern auch den Netzwerkbenutzer `test` vorfinden:

```
$ dscl localhost
 > cd Search
/Search > ls Users/
...
test
...
/Search > ls Mounts/
Tiger-Server.local:/Shares/Users
/Search > quit
Goodbye
```

Abb. 4.28. Der Benutzereintrag des Benutzers *test* ist in der LDAP-Verzeichnisdatenbank gespeichert, das Home-Verzeichnis liegt auf einem AFP-Server.

Ist die Konfiguration korrekt, kann man einen Loginversuch mit dem Netzwerkbenutzer unternehmen. Unter der Voraussetzung, dass eine AFP-Freigabe `Users` mit dem Homeverzeichnis dieses Benutzers auf dem Rechner `Tiger-Server.local` existiert, kann man sich mit dem Namen und dem Kennwort dieses Benutzers anmelden und auf das Homeverzeichnis zugreifen (Abb. 4.28).

A

Systemstart

Der Startvorgang eines modernen Macintosh-Computers beginnt mit einem Selbsttest (Power-On Self Test, kurz POST) und der Initialisierung der Hardware mit Hilfe der Open Firmware. Sofern in dieser Phase keine Fehler auftreten, versucht das System anschließend von dem Laufwerk zu starten, das der Benutzer zuletzt in der Systemeinstellung *Startvolume* definiert hat. Diese Auswahl lässt sich durch das Festhalten der Optionstaste während des Starts überbrücken: Das System stellt dann ein graphisches Auswahlmenü dar, welches die Auswahl des Startlaufwerks erlaubt.

Anschließend lädt das System vom ausgewählten Startlaufwerk die Datei `BootX` aus dem Verzeichnis /System/Library/CoreServices/. BootX lädt den Kernel sowie elementare Treiber. Sobald der Kernel initialisiert und alle nötigen Treiber geladen sind, wird die Root-Partition aktiviert.

Zu diesem Zeitpunkt können die ersten Prozesse gestartet werden. In älteren Mac-OS-X-Versionen wurde diese Aufgabe von den beiden Prozessen `init` (PID 1) und `mach_init` (PID 2) erledigt. In Mac OS X 10.4 gibt es diese beiden Prozesse nicht mehr und `launchd` (PID 1) übernimmt die Rolle des Ur-Vaters aller Prozesse innerhalb des Systems (Abb. A.1).

Während des Startvorgangs liest `launchd` die in den beiden Verzeichnissen /System/Library/LaunchDaemons/ und /Library/LaunchDaemons gespeicherten Konfigurationsdateien und startet die dort definierten Hintergrundprozesse (engl. Daemons). Viele Hintergrundprozesse, die in der Vergangenheit mit Hilfe von Startobjekten (z. B. `mDNSResponder`) oder mittels `xinetd` (z. B. `bootpd`) gestartet worden sind, werden unter Mac OS X 10.4 einheitlich nur noch mit Hilfe von `launchd` gestartet.

Als letzten Schritt startet `launchd` das Programm `loginwindow` (/System/Library/CoreServices), welches die Verantwortung für die Darstellung des Login-Fensters und die Handhabung des Anmeldevorgangs eines Benutzers übernimmt.

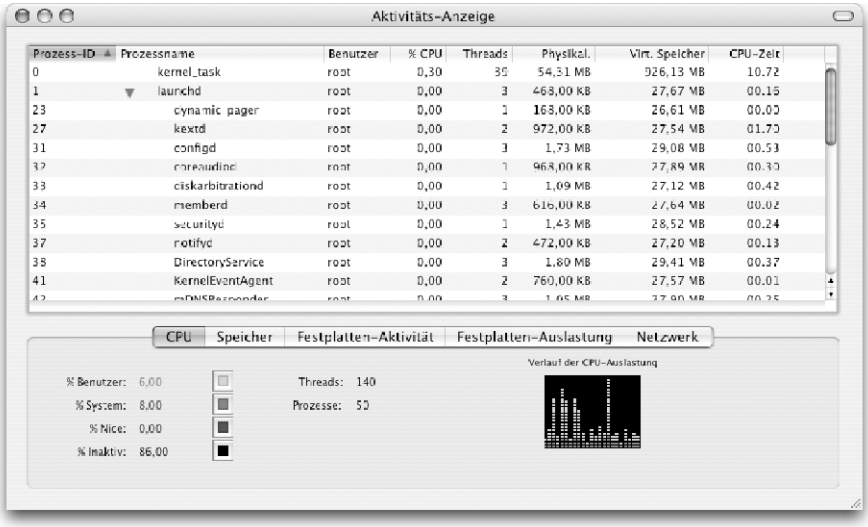


Abb. A.1. In Mac OS X 10.4 übernimmt launchd die Rolle des Ur-Vaters aller Prozesse innerhalb des Systems.

A.1 Launchd und Launch Daemons

Bei den in den Verzeichnissen /System/Library/LaunchDaemons und /Library/LaunchDaemons abgelegten Konfigurationsdateien handelt es sich um XML-Dateien im Property-List-Format.

Während des Systemstarts entnimmt `launchd` diesen Dateien den Ausführungspfad, die Kommandozeilenparameter, die zu registrierenden UDP- oder TCP-Ports und viele andere Informationen. Diese Informationen versetzen `lauchd` in die Lage, die in diesen Dateien beschriebenen Hintergrundprozesse bei Bedarf, also nur wenn diese Prozesse betreffende Anfragen tatsächlich eintreffen, zu starten. Hintergrundprozesse, die weitere Anfragen erwarten, können nach einer bestimmten Zeit beendet werden und damit alle Betriebsmittel freigeben. Sie werden von `launchd` automatisch wieder gestartet, wenn eine erneute Anfrage eintreffen sollte.

`launchd` kombiniert, erweitert und verallgemeinert die Funktionsweise der weiterhin vorhandenen Komponenten `SystemStarter`, `xinetd` und `inetd`.

Um einen Hintergrundprozess mit Hilfe von `launchd` starten zu lassen, muss man zuerst eine entsprechende Konfigurationsdatei im Property-List-Format erstellen und im Verzeichnis /Library/LauchDaemons/ ablegen (/System/Library/LaunchDaemons/ ist den zum Lieferumfang des Betriebssystems gehörenden Konfigurationsdateien vorbehalten).

Die zugelassenen Attribute sind in der man-Page `launchd.plist` dokumentiert. Eine gültige Konfigurationsdatei muss dabei mindestens das Attribut `Label` (eine als Identifikator verwendete Zeichenkette) und das zusam-

mengesetzte Attribut `ProgramArguments` beinhalten. Mit Hilfe des optionalen Boole'schen Attributs `OnDemand` kann man angeben, ob der Hintergrundprozess immer wieder bei Bedarf oder nur einmal gestartet werden muss.

Beim Attribut `ProgramArguments` handelt es sich um ein Array mit Zeichenketten, die den Ausführungspfad und gegebenenfalls notwendige Aufrufparameter enthalten können.

Eine der einfachsten mitgelieferten Konfigurationsdateien beschreibt die Konfigurationsparameter des `mDNSResponders`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Label</key>
        <string>com.apple.mDNSResponder</string>
        <key>OnDemand</key>
        <false/>
        <key>ProgramArguments</key>
        <array>
                        <string>/usr/sbin/mDNSResponder</string>
                        <string>-launchdaemon</string>
        </array>
        <key>ServiceIPC</key>
        <false/>
</dict>
</plist>
```

Der `mDNSResponder` wird nicht immer wieder bei Bedarf, sondern nur einmal gestartet und läuft „unendlich“ weiter. Er unterstützt keine direkte Kommunikation mit `launchd`, weshalb der Wert des Attributs `ServiceIPC` mit `false` angegeben wurde.

A.2 Systemstarter und Startobjekte

Der `SystemStarter` wird von `launchd` mittelbar über das Startskript `/etc/rc` gestartet und kann die in Startobjekten (engl. *StartupItems*) gekapselten systemeigenen oder benutzerdefinierten Dienste starten, stoppen und neu initialisieren.

Aufgrund der in einem solchen Startobjekt gespeicherten Informationen kennt `SystemStarter` die Abhängigkeiten der Startobjekte untereinander und kann die notwendige Reihenfolge selbst bestimmen, während `launchd` alle Dienste bewusst parallel startet.

Natürlich versucht auch `SystemStarter`, möglichst viele Startobjekte auf einmal auszuführen, um den Startvorgang insgesamt zu beschleunigen. Entsprechend der wechselseitigen Abhängigkeiten werden dazu Gruppen von Startobjekten gebildet, welche dann parallel gestartet werden können.

Die Startobjekte selbst werden durch Verzeichnisse mit einem fest definierten Inhalt repräsentiert. Dabei entspricht der Name eines solchen Ver-

zeichnisses dem Namen des so repräsentierten Startobjekts, z. B. das Startobjekt `SSH` aus Mac OS X 10.3:

```
SSH
SSH/Resources
SSH/Resources/da.lproj
SSH/Resources/da.lproj/Localizable.strings
SSH/Resources/Dutch.lproj
SSH/Resources/Dutch.lproj/Localizable.strings
SSH/Resources/English.lproj
SSH/Resources/English.lproj/Localizable.strings
SSH/Resources/fi.lproj
SSH/Resources/fi.lproj/Localizable.strings
SSH/Resources/French.lproj
SSH/Resources/French.lproj/Localizable.strings
SSH/Resources/German.lproj
SSH/Resources/German.lproj/Localizable.strings
SSH/Resources/Italian.lproj
SSH/Resources/Italian.lproj/Localizable.strings
SSH/Resources/Japanese.lproj
SSH/Resources/Japanese.lproj/Localizable.strings
SSH/Resources/ko.lproj
SSH/Resources/ko.lproj/Localizable.strings
SSH/Resources/no.lproj
SSH/Resources/no.lproj/Localizable.strings
SSH/Resources/pt.lproj
SSH/Resources/pt.lproj/Localizable.strings
SSH/Resources/Spanish.lproj
SSH/Resources/Spanish.lproj/Localizable.strings
SSH/Resources/sv.lproj
SSH/Resources/sv.lproj/Localizable.strings
SSH/Resources/zh_CN.lproj
SSH/Resources/zh_CN.lproj/Localizable.strings
SSH/Resources/zh_TW.lproj
SSH/Resources/zh_TW.lproj/Localizable.strings
SSH/SSH
SSH/StartupParameters.plist
```

Eine zentrale Rolle bei der Ausführung eines Startobjekts übernimmt ein Programm – zumeist ein Shell-Skript –, welches den gleichen Namen trägt wie das Startobjekt selbst. Dieses Programm, im obigen Beispiel `SSH/SSH`, wird von `SystemStarter` während des Systemstarts mit dem Argument `start` aufgerufen (wird `SystemStarter` von Hand zu einem späteren Zeitpunkt aufgerufen, dann können auch die Argumente `stop` und `restart` übergeben werden).

Typischerweise importieren Shell-Skripte, die diese Rolle übernehmen, die Datei `/etc/rc.common`, welche eine Reihe nützlicher Funktionsdefinitionen für eigene Startobjekte enthält – u. a. zur Vereinfachung der Behandlung von an das Skript übergebenen Argumenten. Ein einfaches Skript für ein Startobjekt, welches nichts tut (engl. *no operation*, abgekürzt *NOP*) außer einfache Meldungen auszugeben, könnte wie folgt aussehen:

```
#!/bin/sh

##
```

```
# NOP
##

. /etc/rc.common

StartService ()
{
  echo "NOP gestartet"
}

StopService ()
{
  echo "NOP gestoppt"
}

RestartService ()
{
    StopService
    StartService
}

RunService "$1"
```

Die in diesem Skript definierten Funktionen `StartService`, `StopService` und `RestartService` werden von der am Ende des Skripts aufgerufenen Funktion `RunService` aufgerufen. `RunService` ist in der anfangs importierten Datei `/etc/rc.common` definiert, um die Behandlung der möglichen Eingabeargumente `start`, `stop` und `restart` zu erleichtern. Abhängig vom Eingabeargument ruft `RunService` einfach eine der drei Funktionen auf:

```
##
# Generic action handler
##
RunService ()
{
    case $1 in
      start  ) StartService   ;;
      stop   ) StopService    ;;
      restart) RestartService ;;
      *      ) echo "$0: unknown argument: $1";;
    esac
}
```

Um `SystemStarter` die Auswahl des richtigen Startzeitpunkts zu ermöglichen, müssen natürlich weitere Informationen bereitgestellt werden. Die Datei `StartupParameters.plist` enthält dazu eine Reihe von Festlegungen, die von `SystemStarter` gelesen werden, um die korrekte Startreihenfolge zu bestimmen:

`StartupParameters.plist` enthält diese Informationen entweder im Format einer *NeXT Property List* oder im Format einer *XML Property List*, wobei die erste besonders einfach zu lesen ist:

```
{
  Description     = "No Operation";
  Provides        = ("NOP");
```

```
  Requires        = ();
  Uses            = ();
  OrderPreference = "None";
}
```

Diese sehr einfache `StartupParameters.plist` legt weder über das Attribut `Requires` noch über das Attribut `Uses` Beziehungen zu anderen Startobjekten fest und lässt auch sonst über das Attribut `OrderPreference` dem `SystemStarter`-Prozess freie Wahl in Bezug auf die Reihenfolge des Starts.

Bevorzugt wird die Angabe dieser Atribute jedoch im *XML-Property-List* Format. Die gleiche Information wie im obigen Beispiel kann in diesem Format wie folgt dargestellt werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
  "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>Description</key>
    <string>No Operation</string>
<key>Provides</key>
    <array>
    <string>NOP</string>
    </array>
<key>Requires</key>
    <array/>
<key>Uses</key>
    <array/>
<key>OrderPreference</key>
    <string>None</string>
</dict>
</plist>
```

Schlüsselwort	Datentyp	Beschreibung
`Description`	`String`	Eine Kurzbeschreibung des Startobjekts (wird derzeit ignoriert)
`Provides`	`Array`	Eine Liste der vom Startobjekt bereitgestellten Dienste – bestehend zumeist aus genau einem Dienst
`Requires`	`Array`	Eine Liste der vom Startobjekt unbedingt benötigten Dienste. Das Startobjekt wird nicht gestartet, solange nicht alle zu den Diensten aus dieser Liste korrespondierenden Startobjekte gestartet worden sind
`Uses`	`Array`	Eine Liste der vom Startobjekt nicht unbedingt benötigten Dienste. Das Startobjekt muss in der Lage sein, auch dann zu starten, wenn diese Dienste noch nicht gestartet worden sind
`OrderPreference`	`String`	Ein u. U. vom `SystemStarter` ignorierter Hinweis für die Festlegung der Reihenfolge für sonst (im Sinne der durch die Attribute `Requires` und `Uses` festgelegten Beziehungen) gleichberechtigte Startobjekte. Möglich sind `First`, `Early`, `None`, `Late` oder `None`.

Auf diese Weise definierte Startobjekte werden vom `SystemStarter` beim Systemstart immer dann automatisch gefunden und mit dem Argument `start` gestartet, wenn sie entweder im Verzeichnis `/System/Library/StartupItems` oder `/Library/StartupItems/` abgelegt worden sind. Dabei enthält `/System/Library/StartupItems` alle systemeigenen Startobjekte, während `/Library/StartupItems/` für benutzerdefinierte Startobjekte verwendet werden kann.

Nach dem Systemstart können beliebige Startobjekte aus der Kommandozeile heraus – root-Rechte vorausgesetzt – gestartet, gestoppt bzw. neu initialisiert werden:

```
[ibook:~] root# SystemStarter start NOP
Welcome to Macintosh.
NOP gestartet
Startup complete.
Hangup
[ibook:~] root# SystemStarter stop NOP
Welcome to Macintosh.
NOP gestoppt
Startup complete.
Hangup
[ibook:~] root#
```

Die bisher mit einfachen `echo`-Befehlen ausgegebenen Meldungen würden beim grafischen Startvorgang allerdings nicht angezeigt werden. Um Meldungen innerhalb des grafischen Startfensters auszugeben, kann man das Kommando `ConsoleMessage` verwenden, mit dem man Meldungen aus einem Skript heraus zurück an `SystemStarter` übergeben kann.

In unserem Beispielskript müsste also lediglich der Befehl `echo` durch den Befehl `ConsoleMessage` ersetzt werden:

```
#!/bin/sh

##
# NOP
##

. /etc/rc.common

StartService ()
{
  ConsoleMessage "NOP gestartet"
}

StopService ()
{
  ConsoleMessage "NOP gestoppt"
}

RestartService ()
{
    StopService
    StartService
}

RunService "$1"
```

Die Verwendung von `ConsoleMessage` erlaubt uns darüber hinaus zusätzlich die Lokalisierung der ausgegebenen Meldungen. Dazu muss ein Verzeichnis `Resources` innerhalb des Startobjekts angelegt werden, in das wiederum Verzeichnisse für die unterstützten Sprachen anzulegen sind. Darin sind dann Dateien mit dem Namen `Localizable.strings` im XML-Property-List-Format abzulegen, welche die jeweiligen Lokalisierungen der über `ConsoleMessage` ausgegebenen Meldungen enthalten:

```
NOP
NOP/NOP
NOP/Resources
NOP/Resources/English.lproj
NOP/Resources/English.lproj/Localizable.strings
NOP/Resources/German.lproj
NOP/Resources/German.lproj/Localizable.strings
NOP/StartupParameters.plist
```

Um beispielsweise die in unserem Beispielskript ausgegebenen Meldungen auch in englischer Sprache ausgeben zu können, müsste die Datei `NOP/Resources/English.lproj/Localizable.strings` folgenden Inhalt haben:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "file://localhost/System/Library/
  DTDs/PropertyList.dtd">
<plist version="0.9">
<dict>
<key>NOP gestartet</key>
<string>NOP started</string>
<key>NOP gestoppt</key>
<string>NOP stopped</string>
</dict>
</plist>
```

Entsprechend dem Wert der Umgebungsvariablen `LANGUAGE` wählt `SystemStarter` aus den vorhandenen Lokalisierungen dann die passende aus.

B

Produktionseinsatz als Server

Mac OS X verfügt Unix-typisch bereits in der Einzelplatzvariante über viele Leistungsmerkmale, die den Betrieb als Server für zahlreiche Dienste zulassen. Für den Produktionsbetrieb in einem Unternehmen sollte man dennoch zu der Variante *Mac OS X Server* greifen, die für diese Art von Einsatz optimiert wurde und gegenüber der Einzelplatzvariante über zahlreiche Erweiterungen verfügt.

Anders als in der Einzelplatzvariante werden in Mac OS X Server nahezu alle bisher besprochenen Eigenschaften und Netzwerkdienste offiziell unterstützt. Das System wird mit spezifischen graphischen Werkzeugen und zahlreichen zusätzlichen Befehlen für die Kommandozeile ausgeliefert, welche die Konfiguration dieser Dienste dramatisch vereinfachen. Zu den wichtigsten graphischen Werkzeugen zählen dabei der *Arbeitsgruppen-Manager* (engl. Workgroup-Manager) und der *System Admin*. Beide Programme erlauben Fernadministration und können auf einem beliebigen Client-System (also insbesondere auch auf einem PowerBook) installiert werden.

Zahlreiche zusätzliche Komponenten wie z. B. MySQL oder JBoss sorgen für einen gegenüber der Einzelplatzvariante deutlich größeren Funktionsumfang. Der mitgelieferte Open-Directory-Server, eine Kombination von OpenLDAP, einem SASL-basierten Passwort-Server und MIT Kerberos erlaubt den Aufbau von Verzeichnisdienststrukturen, die auch höheren Ansprüchen an Sicherheit und Komfort genügen.

Mac OS X Server wird in einer auf 10 Fileserver-Clients beschränkten und in einer unbeschränkten Version angeboten. Man kann es auf nahezu allen Geräten installieren, die von der Einzelplatzvariante von Mac OS X unterstützt werden. Eine Ausnahme bilden die mobilen Geräte: die Installation von Mac OS X Server auf iBooks und PowerBooks ist bisher zwar praktisch möglich, wird aber nicht offiziell unterstützt oder empfohlen.

Umgekehrt ist Mac OS X Server die einzige Variante, die sich auf den Apple-Servern vom Typ *Xserve* installieren lässt. Xserves werden mit der unbeschränkten Version von Mac OS X Server ausgeliefert und sind in den

meisten Fällen die Plattform der Wahl für den Betrieb eines Mac-OS-X-basierten Produktionsservers.

Beim Xserve handelt es sich um einen dedizierten 1 HE (Höheneinheit), Server, der für den platzsparenden Einbau in ein 19″-Standardrack und Dauereinsatz konzipiert wurde. Die aktuelle Variante Xserve G5 verfügt über einen oder zwei 64-Bit-IBM-Prozessoren vom Typ PowerPC 970, drei Einschübe für S-ATA-Laufwerksmodule, bis zu 16 GB Hauptspeicher sowie zahlreiche Sensoren, die den Betriebszustand der Hardwarekomponenten im laufenden Betrieb überwachen. Zum Lieferumfang von Mac OS X Server gehört das graphische Werkzeug *Server Monitor*, mit dem man die Signale dieser Sensoren auswerten und graphisch darstellen kann.

B.1 Arbeitsgruppen-Manager

Mit dem Arbeitsgruppen-Manager lassen sich Benutzer- und Gruppen (Abb. B.1), Zugriffsrechte und Freigaben (Abb. B.2) verwalten. Darüber hinaus kann man mit Hilfe des Arbeitsgruppen-Managers Computereinträge und Standardvoreinstellungen für bestimmte Benutzer oder Benutzergruppen vorgeben (Managed Client / MCX Preferences).

Computereinträge kann man mit Computerlisten gruppieren und gemeinsam verwalten. Auf diese Weise kann man den Zugriff auf bestimm-

Abb. B.1. Die Verzeichniseinträge für Benutzer und Gruppen lassen sich im Arbeitsgruppen-Manager anzeigen und editieren.

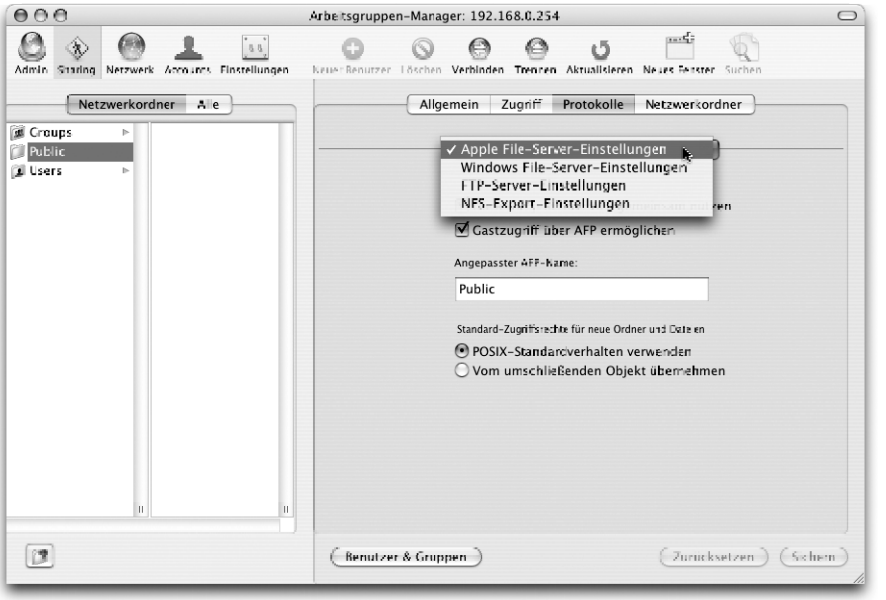


Abb. B.2. Im Arbeitsgruppen-Manager kann man komfortabel AFP-, CIFS-/SMB-, FTP- und NFS-Freigaben einrichten.

te Computer reglementieren oder bestimmten Computern bestimmte MCX-Voreinstellungen zuweisen (z. B. räumlich nah aufgestellte Drucker als Standarddrucker definieren) (Abb. B.3 und Abb. B.4).

Bei den Konfigurationsdaten des Arbeitsgruppen-Managers handelt es sich dabei um Einträge in Verzeichnisdiensten, die er über Open Directory liest und schreibt. Dabei spielt es keine Rolle, ob es sich beim verwendeten Verzeichnisdienst um einen LDAP-Server, einen Active-Directory-Server oder um eine NetInfo-Datenbank handelt. Man kann mit dem Arbeitsgruppen-Manager sogar die lokale NetInfo-Datenbank öffnen und die lokalen Benutzer- bzw. Gruppeneinträge sichten und bearbeiten[1].

Der Arbeitsgruppen-Manager kann demnach mit gewissen Einschränkungen als Editor für Verzeichnisdaten betrachtet werden. Seit Mac OS X Server 10.3 gibt es im Arbeitsgruppen-Manager eine Detailansicht, in der man die Attribute und Werte von Verzeichniseinträgen auch im Textformat (also quasi im Rohformat) ansehen und editieren kann (Abb. B.5).

Die in der Detailansicht dargestellten Daten sind eine Meta-Ansicht auf der Basis der von Open Directory durchgeführten Abbildung der nativen Verzeichnisattribute auf ihre Open-Directory-Entsprechungen (vgl. z. B. `man DirectoryServiceAttributes`). Die zur in Abb. B.5 dargestellten De-

[1] Insbesondere kann man auf diese Weise auch direkt auf dem Client MCX Voreinstellungen definieren!

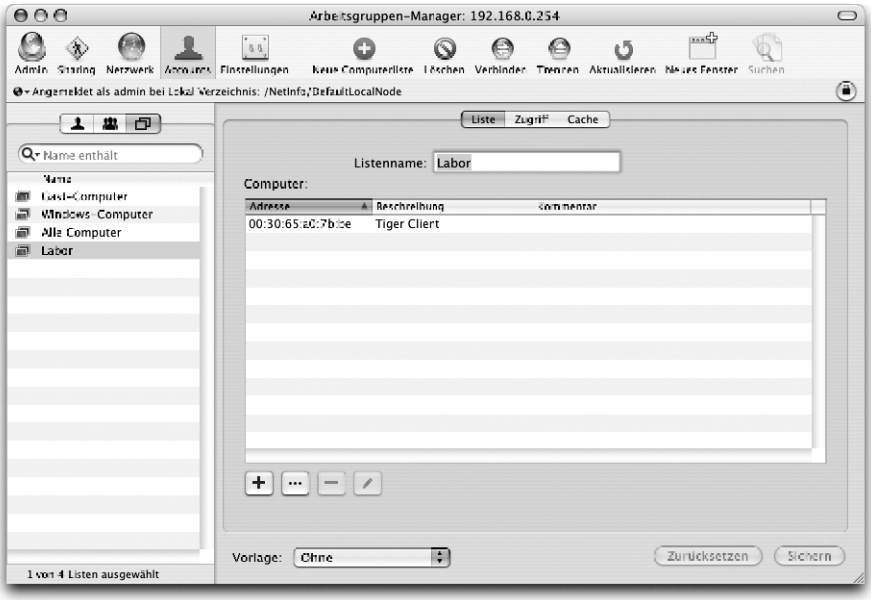


Abb. B.3. Computereinträge kann man mit Computerlisten gruppieren und gemeinsam verwalten.

Abb. B.4. Standardvoreinstellungen (MCX Preferences) kann man einzelnen Benutzern, Benutzergruppen oder Computerlisten zuweisen.

Abb. B.5. In der Detailansicht sieht man die Attribute und Werte der Verzeichniseinträge.

tailansicht passenden „echten“ Daten liegen in der lokalen NetInfo-Datenbank des Servers und können dementsprechend mit `nicl` ausgelesen werden:

```
$ nicl . -read /users/admin
name: admin
home: /Users/admin
gid: 20
picture: /Library/User Pictures/Animals/Butterfly.tif
uid: 501
hint:
sharedDir:
shell: /bin/bash
passwd: ********
authentication_authority: ;ShadowHash;
realname: Administrator
generateduid: E0D5BAE0-EF0C-468B-9B1B-318C059C6385
mcx_flags: <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>has_mcx_settings</key>
        <false/>
</dict>
</plist>

_writers_passwd: admin
_writers_tim_password: admin
_writers_picture: admin
```

```
_writers_hint: admin
_shadow_passwd:
_writers_realname:
```

B.2 Server Admin

Mit der Anwendung *Server Admin* kann man den Status der aktiven Netzwerkdienste überwachen, Protokolldateien einsehen, Konfigurationseinstellungen vornehmen sowie einzelne Dienste starten und stoppen (Abb. B.6).

Server Admin bietet eine graphische Benutzeroberfläche, mit der sich die den Diensten zugrunde liegenden Komponenten auf einfache Weise konfigurieren lassen (Tabelle B.1).

Um nun beispielsweise die Apache-Konfiguration zu ändern, muss man unter Mac OS X Server nicht mehr die Konfigurationsdatei `httpd.conf` editieren, sondern kann auf die in Server Admin integrierte graphische Benutzeroberfläche zurückgreifen (Abb. B.7).

Entsprechende Kenntnisse vorausgesetzt, kann man jedoch in den allermeisten Fällen ebenso gut die entsprechende Konfigurationsdateien editieren:

```
$ cat /etc/httpd/httpd.conf
#### Default httpd.conf for Mac OS X Server
####
#### This httpd.conf differs from the httpd.conf distributed
```

Abb. B.6. Mit der Anwendung *Server Admin* kann man den Status der aktiven Netzwerkdienste überwachen, Protokolldateien einsehen, Konfigurationseinstellungen vornehmen sowie einzelne Dienste starten und stoppen.

Tabelle B.1. Server Admin bietet eine graphische Benutzeroberfläche, mit der sich die den Diensten zugrunde liegenden Komponenten auf einfache Weise konfigurieren lassen.

Server Admin Dienst	Hauptkomponenten
AFP	AppleFileServer
Anwendungsserver	JBoss, Tomacat
DHCP	bootpd
DNS	BIND
Firewall	Kernel, ipfw
FTP	xftpd
iChat	Jabber
Mail	Postfix, Cyrus, SpamAssassin, ClamAV, Mailman, SquirrelMail
NAT	ipfw, natd
NetBoot	bootpd
NFS	nfsd, rpc.lockd, rpc.statd, portmap
Open Directory	OpenLDAP, Berkeley DB, PasswordServer, MIT Kerberos
Print	CUPS
QuickTime Streaming	QuickTime Streaming Server
Software Aktualisierung	swupd, swupd_syncd
VPN	Racoon
Web	Apache
WebObjects	WebObjects
Windows	Samba
Xgrid	xgridagentd, xgridcontrollerd

```
#### with Apache and the httpd.conf present on Mac OS X.
#### Feel free to edit this; the Server Admin app also edits this
#### file but will respect your changes unless noted below.
#### See also ReadMe.txt.
...
```

Die Anwendung Server Admin kommuniziert auf der Serverseite mit dem Hintergrundprozess `servermgrd`. Dieser Daemon basiert auf Apache und führt auf der Serverseite die Anweisungen von Server Admin aus. Die mit Hilfe von SSL gesicherte Kommunikation erfolgt über den Port 311:

```
$ sudo lsof +c 0 -i:311
COMMAND    PID  USER   FD   TYPE     DEVICE SIZE/OFF NODE NAME
servermgrd  65  root    6u  IPv4 0x02ee2dd0      0t0  TCP *:asip-webadmin
                                                        (LISTEN)
```

`servermgrd` verfügt über eine rudimentäre Weboberfläche, mit der sich die unterstützten Dienste ansteuern lassen. Auf diese Weise lassen sich die Serverdienste im Prinzip von einem beliebigen Browser aus fernsteuern (Abb. B.8 und Abb. B.9).

Alternativ steht der Befehl `serveradmin` zur Verfügung, mit dem man alle Funktionen von Server Admin auch in der Kommandozeile (ggf. remote

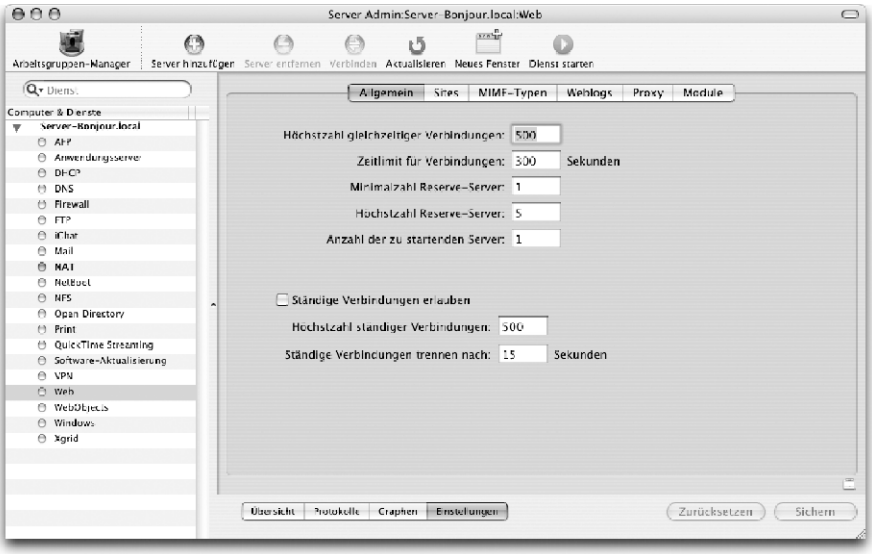


Abb. B.7. Server Admin beinhaltet eine graphische Benutzeroberfläche für die Bearbeitung der Konfiguration des Apache-Webservers.

über eine SSH-Verbindung) ausführen kann. Die Liste der Dienste entspricht im Wesentlichen den in Server Admin dargestellten Diensten:

```
$ sudo serveradmin list
afp
appserver
dhcp
dirserv
dns
filebrowser
ftp
info
ipfilter
jabber
mail
nat
netboot
network
nfs
print
privs
qtss
qtsscontents
signaler
smb
swupdate
vpn
web
webobjects
xgrid
xserve
```

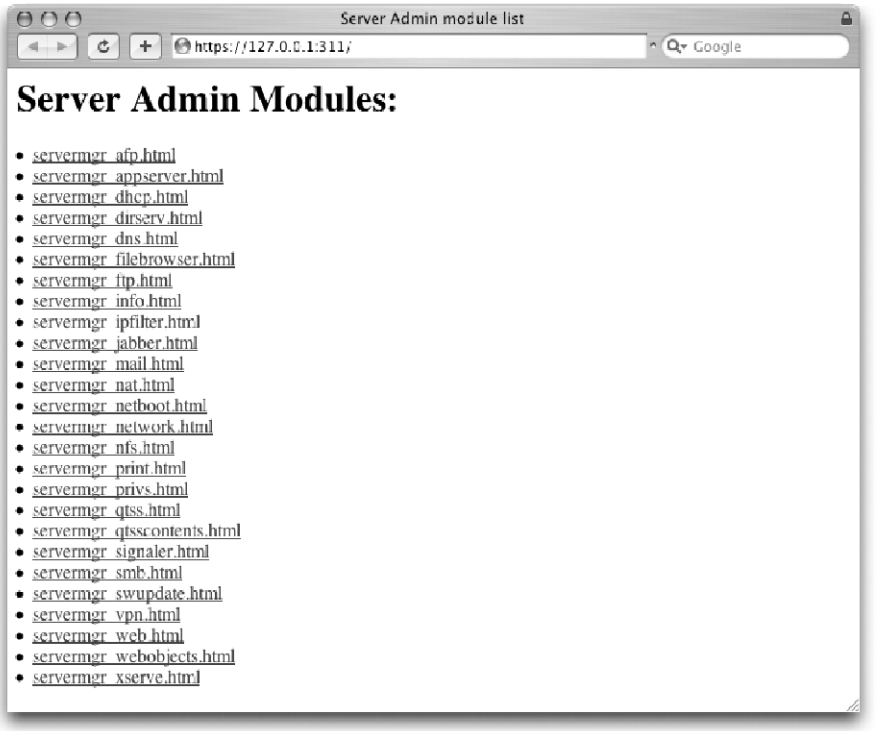


Abb. B.8. Sämtliche Serverdienste können auch in der Weboberfläche angesteuert werden.

Der Dienst `xserve` erlaubt den Zugriff auf die Sensorik eines Xserves und bildet die Funktionalität der Anwendung Server Monitor auf der Kommandozeilenebene ab. Mit einfachen Anweisungen lassen sich Dienste starten und stoppen:

```
$ sudo serveradmin start afp
afp:state = "RUNNING"
afp:status = 0
```

Für Konfigurationseinstellungen verwendet `serveradmin` für alle Dienste ein einheitliches Ein- und Ausgabeformat, was automatisierte Weiterverarbeitung und Konfiguration mit Hilfe von Skripten stark vereinfachen kann:

```
$ sudo serveradmin settings afp
afp:adminGetsSp = no
afp:activityLogTime = 7
afp:registerAppleTalk = yes
afp:afpTCPPort = 548
afp:createHomeDir = yes
afp:lock_manager = yes
afp:autoRestart = yes
afp:noNetworkUsers = no
```

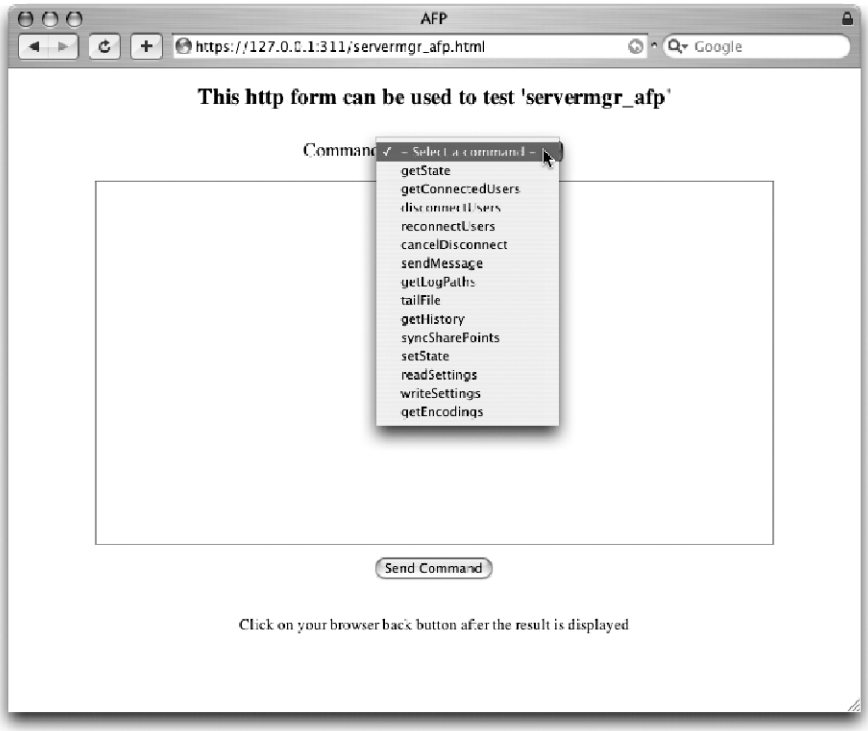


Abb. B.9. Jeder Dienst verfügt über eine Reihe von Methoden zur Steuerung und Statusabfrage.

```
afp:TCPQuantum = 262144
...

$ sudo serveradmin settings smb
smb:max smbd processes = 0
smb:encrypt passwords = yes
smb:allow trusted domains = no
smb:printer admin = "@admin, @staff"
smb:wins support = no
smb:brlm = "yes"
smb:deadtime = 5
smb:display charset = "UTF-8-MAC"
smb:server string = "Mac OS X"
...
```

B.3 Weitere Werkzeuge und Dokumentation

Zum Lieferumfang von Mac OS X gehören zahlreiche weitere Werkzeuge. Mit Hilfe des Befehls `sharing` kann man z. B. auf einfache Weise Freigaben für AFP, SMB/CIFS und FTP einrichten. Mit dem `diskspacemonitor`

Abb. B.10. Die vollständige Dokumentation zu Mac OS X Server steht bei Apple im PDF-Format zum freien Download zur Verfügung.

lässt sich der freie Festplattenplatz überwachen und Alarm beim Überschreiten bestimmter Grenzen auslösen. Und mit `systemsetup` und `networksetup` stehen Werkzeuge zu Verfügung, mit denen sich die Konfiguration des Servers im Detail festlegen lässt.

Darüber hinaus stehen zahlreiche dienstspezifische Werkzeuge zur Verfügung: z. B. `slapconfig`, `NeST`, `kdcsetup`, `sso_util` oder `mkpassdb` für die manuelle Konfiguration eines Open Directory Servers.

Die vollständige Dokumentation zu Mac OS X Server findet man unter (Abb. B.10):

```
http://www.apple.com/de/server/documentation/
```

C

Literaturhinweise

C.1 Netzwerk – Grundlagen

Sidhu, Gursharanet S. et al.: *Inside AppleTalk*. Addison-Wesley, 1990.

Eine sehr detaillierte Beschreibung der AppleTalk-Protokollfamilie. Obwohl AppleTalk zunehmend an praktischer Bedeutung verliert, ein sehr lesenswertes Buch – gerade für diejenigen, die TCP/IP bereits im Detail kennen und sich für ZeroConf/Bonjour interessieren.

Stevens, Richard W.: *TCP/IP Illustrated, Volume 1, The Protocols*. Addison-Wesley, 1994.

Eine hervorragende Einführung in die Basisfunktionalität der TCP/IP-Protokollfamilie. Anhand von vielen Abbildungen und unter Zuhilfenahme von Werkzeugen wie `tcpdump` wird die Funktionsweise von IP, ARP, RARP, ICMP, UDP und TCP genauestens erklärt. Die meisten Beispiele basieren auf Standardbefehlen und lassen sich mit wenig Aufwand unter Mac OS X nachvollziehen. Statt des in den Übungen verwendeten Befehls `sock` kann unter Mac OS X der Befehl `nc` (netcat) verwendet werden.

C.2 Netwerk – Anwendungen

Albitz, Paul und Liu, Cricket: *DNS and BIND*. O'Reilly, 2001.

Eine sehr praxisorientierte Darstellung von DNS an Hand von BIND mit detaillierten Konfigurationsbeispielen für nahezu alle denkbaren Anwendungsfälle. Da BIND zum Lieferumfang von Mac OS X gehört, ist der Inhalt dieses Buches nahezu ohne Einschränkungen auf Mac OS X übertragbar.

Apple Computer, Inc.: *Apple Filing Protocol Version 3.1*, 2003.

An Entwickler gerichtete Dokumentation der aktuellen Version des AFP-Protokolls. Detaillierte Beschreibung der Funktionsweise von AFP inkl. einer

Auflistung aller Fehlercodes. Letzteres ist insbesondere sehr nützlich für die Interpretation der Fehler- und Zugriffsprotokolle des Mac OS X AFP-Servers.

Ts, Jay et al.: *Using Samba*. O'Reilly, 2003.

Beschreibt die Funktionsweise von SMB/CIFS am Beispiel der freien Implementierung Samba. Die Konfigurationsbeispiele sind sehr gut auf Mac OS X übertragbar. Das Buch selbst ist in der Standardinstallation von Mac OS X als HTML enthalten[1].

C.3 Verzeichnisdienste

Apple Computer, Inc.: *Understanding and Using NetInfo*, 2001.

War Bestandteil der Dokumentation von Mac OS X Server 10.2. Beschreibt die grundlegenden Konzepte von NetInfo und bietet einen ersten Einstiegspunkt in diese Thematik.

Carter, Gerald: *LDAP System Administration*. O'Reilly, 2003.

Eine gute LDAP-Einführung mit zahlreichen Konfigurationsbeispielen für OpenLDAP. Da OpenLDAP in Mac OS X integriert ist, lassen sich die Konfigurationsbeispiele auch hier ohne Probleme auf Mac OS X übertragen.

Xedoc Software Development Pty. Ltd.: *NetInfo Editions 4.x User Manual*, 1997.

Handbuch zu einer NetInfo-Implementierung für Nicht-NeXT-Systeme von Xedoc. Eine an die Administratoren von Unix-basierten Systemen gerichtete, sehr praxisorientierte und ausführliche Einführung in die Konzepte und in die manuelle Konfiguration von NetInfo mit Hilfe der Kommandozeile. Die Konfigurationsbeispiele lassen sich gut auf Mac OS X übertragen.

[1] /usr/share/swat/using_samba/