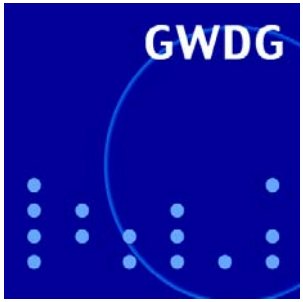


GWDG-Bericht Nr. 59

Dirk von Suchodoletz

Effizienter Betrieb großer Rechnerpools

Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Dirk von Suchodoletz

Effizienter Betrieb großer Rechnerpools

Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen

Dirk von Suchodoletz

Effizienter Betrieb großer Rechnerpools

Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen

GWDG-Bericht Nr. 59
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen
Am Faßberg
D-37077 Göttingen
Telefon: 0551-201-1510
Telefax: 0551-21119
E-Mail: gwdg@gwdg.de

Satz: Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Druck: Offset- und Dissertations-Druck Jürgen Kinzel, Göttingen-Weende

ISSN 0176-2516

Effizienter Betrieb großer Rechnerpools

Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen

Dirk von Suchodoletz
Göttingen 2002

Universität Göttingen - GwDG
Projektbericht:

Effizienter Betrieb großer Rechnerpools

Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen

Dirk von Suchodoletz
Kreuzbergring 56
37075 Göttingen
dirk@goe.net

4. Juli 2002

Inhaltsverzeichnis

Teil I

Vorbetrachtung und Problemstellung

Kapitel 1

Einleitung

Während sich die einen von der Informationsrevolution wachsenden Wohlstand versprechen, gilt sie anderen als Vernichterin von Arbeitsplätzen. Dass die Informationstechnik die bedeutendste Produktivkraft der Gegenwart oder sogar der Geschichte sei, gilt quer durch alle Lager als Selbstverständlichkeit, kritische Stimmen wollen trotzdem nicht verstummen. Von Stephen Roach, dem Chefökonomen der Investmentbank Morgan Stanley und dem Nobelpreisträger der Ökonomie Robert Solow ist die Bemerkung überliefert, *man könne das Computerzeitalter überall sehen, außer in der Produktivitätsstatistik*[1].

Nach dem Hype der letzten Jahre zieht in die Welt der Computernetze mittlerweile der Druck von realen Organisationsproblemen und realistischen Lösungskonzepten ein. Einen Zwang, den die IT-Branche auf ihrem Hochgeschwindigkeits-Siegeszug durch alle betrieblichen und organisatorischen Abläufe hindurch bislang kaum kennenlernen mußte. Angesichts der gegenwärtigen Krisen der erfolgsverwöhnten Branche steht zu fragen, ob die Suche nach den innovativen Konzepten von morgen einzig im alten "höher, schneller, weiter" (und teurer) der vergangenen Dekade liegen kann, oder ob nicht vielmehr die eigentlichen Revolutionen erst unter gewissem Problemdruck und im Angesicht faktischer Problemstellungen zustande kommen.

Die Ausgangssituation ist definiert: Seit den 70er Jahren dringt die Informationstechnik, ermöglicht durch die Fortschritte der Mikroelektronik und der Informatik, in immer mehr Bereiche der Volkswirtschaft und des öffentlichen Lebens ein und seit den 80er Jahren finden sich an immer mehr Arbeitsplätzen ein Personal-Computer (PC). Diese sind in den überwiegenden Fällen in ein lokales Rechnernetzwerk integriert und viele Arbeitsplätze haben darüber hinaus Zugriff auf die verschiedenen Dienste des Internets. Vernetzte

[1] siehe [Z2], Seite 100

Rechner haben die Aufgabe, nicht nur die Arbeitsproduktivität zu steigern und ihre Benutzer von stupiden und sich wiederholenden Aufgaben zu entlasten, sondern sie bieten vor allem enorme Potentiale in der Reorganisation betrieblicher Abläufe und helfen, Produktionsfelder und Dienstleistungen neu zu definieren oder zu erschließen.
Die Anzahl der Dienste, die von Rechnernetzen erbracht werden sollen und die Qualitätsanforderungen an die unterschiedlichen Dienste seitens der Benutzer und Betreiber von Rechnernetzen erfahren hierbei eine stetige Ausweitung. Ein Wachstum ist auch bei der Ausdehnung von Rechnerarbeitsplätzen zu verzeichnen, wodurch die Aufgabe des Betriebes dieser Netze immer umfangreicher wird. Neben Personal, das Wartungs- und Routinearbeiten an Rechnern, ihren Komponenten und im Netzwerk durchführt, erfordert der Betrieb größerer Rechnerpools eine ständig steigende Anzahl von Experten. Dem Arbeitsmarkt stehen heute ausreichend viele Personen zur Verfügung, die entweder über Grundkenntnisse in der Rechneradministration oder Computerhardware verfügen oder sich entsprechend aus- und weiterbilden lassen. Es herrscht jedoch ein großer Mangel an Fachleuten zum Betrieb von Rechnernetzen, der in absehbarer Zeit nicht behoben werden kann, da die stetig sich verändernde Technik und die wachsenden Anforderungen an Rechnernetze und die von ihnen erbrachten Dienste neben einer umfassenden Ausbildung auch eine ständige Weiterbildung sowie einen großen Erfahrungsschatz erfordern.
Aus Gründen der Kostenersparnis erfreuen sich in den vergangenen Jahren Software-Werkzeuge einer wachsenden Beliebtheit, die das Personal bei den Routinearbeiten des Betriebs von Rechnernetzen unterstützen.
In dieser Arbeit soll es darum gehen, ein Frameset zur Automatisierung des Betriebes größerer Rechnerzahlen vorzustellen und einige Beispiele sogenannter Managementwerkzeuge zu präsentieren, die im wesentlichen den Zugriff auf Informationen über die zu betreibenden Rechnern erleichtern und die Änderung von Konfigurationen der verschiedenen Systeme erlauben. Der Bedarf an Fachleuten kann hierdurch reduziert werden, da sich der Personalaufwand für Routineaufgaben verringern läßt. Weiterhin besteht das Ziel, bei Wachstum, Anpassung und Erweiterungen des Rechnernetzwerkes den Personalbedarf nicht linear mit der Zahl der eingesetzten Maschinen steigen zu lassen.
Den Hintergrund für das im folgenden vorgestellte Projekt bildet eine Problemstellung unter keineswegs einfachen organisatorischen Bedingungen und (für Computerexperten) sicherlich äußerst restringierten Ressourcen: Die Schaffung eines Computernetzes mit mehreren hundert verschiedenartig konfigurierten Rechnern für unterschiedliche Anwendungen und Anwender der Georg-August-Universität Göttingen - mit der Einschränkung, nur geringe

Mittel für Stellen und die Anschaffung neuer Soft- und Hardware zu haben. Die knappen Ressourcen schärfen den Blick für das Mach- und Verwaltbare und generieren möglicherweise Lösungen, bei denen die Abschätzungen eines gemeinsamen Nutzens (im vorliegenden Fall für die angeschlossenen Studierenden und Institute der Universität Göttingen) weit maßgeblicher für die entwickelten Lösungen sind und sein mußten.
Die Hoffnung des Projekts, eine in mancher Hinsicht vorbildliche Struktur vernetzter Rechner entwickelt zu haben, bildet sich nicht nur aus dem Erreichten, sondern auch aus der geschaffenen Offenheit und Erweiterungsfähigkeit des gefundenen Konzepts, die auch über den universitären Mangelverwaltungsbereich bzw. Improvisationsbereich[2] hinweg tragfähig sein sollte.
In dieser Arbeit wird es darum gehen, die Strukturen der Lösung in ihrer Entstehungsgeschichte nachzuzeichnen und die zentralen Installations- und Anwendungsmerkmale incl. ihrer Datenbanksteuerung nachvollziehbar zu dokumentieren. Obwohl es stärkere Differenzierungen in den Aufgabenfeldern an einzelnen Arbeitsstationen geben kann, ist es in vielen Fällen jedoch möglich, eine recht große Plattform an Gemeinsamkeiten zwischen verschiedenen Ausstattungen zu finden.

[2] Dieser Bereich hat durchaus auch den Charakter von Forschung, wenn jedoch auch ein wichtiger Punkt hierfür, die gute Personal- und Materialausstattung, selten erfüllt sind.

Kapitel 2

Problemstellung

2.1 Projektziele

Das Ziel dieser Arbeit besteht in der Beschreibung eines Konzeptes zum Betrieb einer größeren Zahl von Rechnerarbeitsplätzen mit Internet-Zugang. Hierbei geht es um den Aufbau und die Verwaltung dieser Arbeitsplätze unter Berücksichtigung verschiedener Aspekte: Die Arbeitsplätze sollen einem größeren heterogenen Benutzerkreis an dezentralen Punkten zur Verfügung stehen. Aus Gründen der Administrierbarkeit und Transparenz für die Benutzer wird eine einheitliche Software-Plattform angestrebt, die ein leichtes Zurechtfinden an jedem dieser Rechner gestattet. Trotzdem soll dabei die Auswahl der Software nicht eingeschränkt werden, um vielfältigen und zukünftigen Anforderungen gerecht werden zu können. Dies bedeutet, eine Plattform zu entwickeln, die eine größtmögliche Auswahl an Programmen zur Verfügung stellen kann. Der resultierende Verwaltungsaufwand für diese Programme soll jedoch so gering wie möglich gehalten werden.
Diese Arbeit beschäftigt sich deshalb mit einem Lösungskonzept zur Steuerung, Überwachung, Verwaltung und Administration größerer Rechnerpools. Das vorzustellende Konzept beinhaltet die Beschreibung einer Betriebssystemplattform, auf der die obengenannten Anforderungen erfüllt werden können. Hierbei geht es jedoch nicht um die Einrichtung einzelner Software-Pakete für die konkrete Nutzung, sondern um die notwendigen Grundlagen, um ein einfaches Management dieser Pakete zu ermöglichen. Für die Auswahl des einzusetzenden Betriebssystems ist die Verfügbarkeit von Anwendungen dabei genauso mitentscheidend wie die Fähigkeiten des Betriebssystems, diese Anwendungen möglichst aufwandsarm zu verwalten.
Von vornherein kann die angestrebte Lösung nicht auf eine geringe Zahl von Arbeitsplätzen beschränkt sein, sondern muss einer steigenden Nachfrage

Rechnung tragen können. Dies bedeutet, dass der Einzelaufwand pro Arbeitsplatz bei steigender Zahl sinken muss, um eine gleichbleibende Betreuungsqualität bei konstantem Mitarbeiterstand erreichen zu können. Eine Steigerung des Personalaufwandes sollte höchstens im Umfang des höheren Hardwareeinsatzes erfolgen, wobei dieser selbst sparsam zu gestalten ist. Dies bedeutet zum einen hohe Lauf- und Lebenszeiten einmal angeschaffter Systeme und zum anderen die Reduktion der klassischen Workstation auf notwendige, wartungsarme Komponenten ohne den Benutzungskomfort einzuschränken. Alle gängigen Medienformen[1] sollten vom Benutzer einsetz- und abrufbar sein. Dabei besteht jedoch immer eine Abhängigkeit zwischen dem (technologischen) Alter der Hardware und ihrer gewünschten Einsatzzeit und den Möglichkeiten der Medienwiedergabe. Deshalb wird es unterschiedliche Gerätetypen mit verschiedenen Leistungsmerkmalen geben, die jeweils in ihrer Geräteklasse den optimalen Einsatz erlauben sollten.
Der wichtigste Reduktionspfad des Administrationsaufwandes liegt in der hohen Standardisierung der Plattformen, was ein gewisses Maß an Abstraktion der gegebenen Hardware voraussetzt. Eine weitere Variante liegt in der möglichst hohen Zentralisierung aller wesentlichen Aufgaben und Server. Die einmal geschaffene Zentralisierung kann die Grundlage weiterer Anknüpfungspunkte über das ursprünglich definierte Ziel hinaus bieten. Dies betrifft hier zum einen Aspekte der Komponentenverwaltung und der Verteilung von Investitionen. Zum anderen geht es um verschiedenartige Aufgaben der Funktionsüberwachung. Dezentralität, die bei einer großen Anzahl eingesetzter PC's häufig als Hindernis empfunden wird, soll sich durch vereinfachtes Management und bessere Überwachbarkeit nicht mehr als Nachteil auswirken.
Daneben muß im Bereich netzwerkbasierter Arbeitsplätze Sichheitsaspekten Rechnung getragen werden: Wie können die Systeme wirkungsvoll vor Angriffen aus dem eigenen Netz bzw. dem Internet, dessen Bestandteil sie zwangsläufig sind, wirkungsvoll geschützt werden? Wie kann ein hohes Maß an Sicherheit mit möglichst geringen Einschränkungen des Komforts erreicht werden?
Schließlich müssen Backupstrategien, Fail-Over-Lösungen, Sicherheit von Benutzerdaten in der Betrachtung berücksichtigt werden. Wie kann hohe Redundanz der Software, aber auch der eingesetzten Hardware erreicht werden? Der Aufwand des Austauschs von Geräten und Komponenten muss gering bleiben und ohne aufwändige Neukonfigurtation erfolgen können.
Die durch technische und Marktentwicklung zwangsläufig divergierende Aus-

[1] meint "Multimedia", d.h. über die reine Text- und Grafikdarstellung hinausgehende Gestaltungs- und Kommunikationsmittel, min. Audiounterstützung und evtl. Ausgabemöglichkeit diverser Bild- und Videoformate

stattungen sollen auf eine einheitliche Basis gestellt und verwaltet werden können, ohne sich dabei am kleinsten gemeinsamen Nenner orientieren zu müssen. Die angestrebte Architektur soll offen genug ausgelegt sein, um zukünftige Entwicklungen und Tendenzen des Betriebes rechnergestützter Arbeitsplätze subsummieren zu können. Vorteilhaft wäre letztendlich eine Plattform, welche parallel im Hintergrund für wissenschaftliche Anwendungen, wie verteiltes Rechnen etc. genutzt werden könnte.

2.2 Komponenten/Umfang der Lösung

Aus dem geschilderten Szenario ergibt sich eine Reihe von Anforderungen, die sich in fünf größere Bereiche untergliedern lassen.

- Aufbau des Desktops
- Netzwerksoftware
- Hardware der Rechner
- Netzwerkhardware und Infrastruktur
- Verwaltung und Monitoring

Unter dem ersten Punkt liegt der Schwerpunkt in der Begründung einer geeigneten Plattform, welche alle gängige Software und Netzwerkprotokolle unterstützt. Dazu gehört die Verfügbarkeit geeigneter Officeprodukte, Clients für diverse Web-Browser, Text- bzw. multimediabasierte Kommunikationsdienste und ein großer Umfang an verfügbaren Programmiersprachen verschiedenster Auslegung sowie Java-Umgebungen und evtl. verfügbare Emulatoren anderer Hardwareplattformen oder Software-Architekturen.
Die Netzwerksoftware sollte integraler Bestandteil des verwendeten Betriebssystems sein und keinen gesonderten Anpassungsaufwand erfordern. Die eingesetzten Protokolle sollten offengelegt[2] und ausreichend implementiert sein. Open-Source-Software erleichtert in vielen Fällen die Umsetzung, da sie o.g. Kriterien erfüllt.
Ein wesentlicher Aspekt bei der Auswahl der Rechnerhardware liegt in den Kosten der Einzelkomponenten oder Komplettsysteme. Eine Orientierung an der am Massenmarkt verfügbare Hardware trägt wesentlich zur Reduzierung des Anschaffungs- und Ersatzbeschaffungsaufwands bei. Plötzliche Richtungswechsel seitens der Produzenten oder die Einstellung ganzer Produktlinien werden hier unwahrscheinlicher oder vollziehen sich in längeren

[2] Dieses garantiert die Weiterentwicklung und Verlässlichkeit der dauerhaften Verfügbarkeit

Zeiträumen. Teile sollten hochstandardisiert und deshalb leichter austauschbar sein. Abhängigkeiten von einzelnen Herstellern oder wenigen Händlern sollen vermieden werden.
In vielen Fällen sind die Entscheidungen der Netzwerkhardware, d.h. der aktiven Komponenten und der Leitungsgeschwindigkeiten bereits gefallen. Auch hier spielt auf der Rechnerseite Massenhardware eine Rolle. Bei der Auswahl der aktiven Komponenten kommt es wiederum auf die Preisrelation zwischen Administrations- und Überwachungsaufwand, sowie die Sicherheit des Betriebes an. Diese Aspekte können jedoch nur am Rande behandelt werden, wo sie eine unmittelbare Rolle bei der Implementierung der Gesamtlösung spielen.

2.3 Restriktionen im speziellen Fall: Universität

An der Universität Göttingen sollte zur Unterstützung von Forschung und Lehre die Zahl der Internetrechner, die Studierenden zur Verfügung gestellt werden können, stark ausgebaut werden. Das Vorhandensein solcher Arbeitsplätze bildet inzwischen eine der Grundlagen einer fundierten universitären Ausbildung. Eine ganze Palette von Angeboten, angefangen vom elektronischen Vorlesungskommentar, über Skripten, Beispiele und Aufgaben bis hin zum Zugriff auf elektronische Recherchesysteme der verschiedenen Bibliotheken, steht inzwischen über das Internet bzw. ein campusweites Intranet zur Verfügung.
Wie jedoch im universitären Umfeld üblich stehen nur begrenzte finanzielle Mittel zur Verfügung. Das in diesem Projekt gebundene Personal sollte nicht übermäßig aufgestockt werden, da sich der Bereich der EDV-Versorgung in einer Organisation wie einer Universität dynamisch verändert. Die Flexibilität einer solchen Betriebsgruppe ist auch von ihrer Größe abhängig, die deshalb bestimmte Grenzen nicht überschreiten sollte. Meistens spielen noch Randaspekte, wie die räumliche Unterbringung und die zur Verfügung stehende Gesamtfläche an Maschinenstell- und Bürofläche eine Rolle. Dezentralität ist für die Erreichbarkeit der Arbeitsplätze seitens der Benutzer wichtig, sollte aber im Bereich der Administration vermieden werden, um den Kommunikations-Overhead nicht ausufern zu lassen[3].
Die Aufgabenstellung läßt sich in folgende Einzelforderungen unterteilen:

- Schaffung eines Pools robuster Arbeitsplatzrechner ohne viele mecha-

[3]Die organisations- und kommunikationstheoretische Aspekte die beim Design der Lösung relevant waren, gehen nicht weiter in die hier vorgestellten Überlegungen ein, da sie den Rahmen der vorgelegten Arbeit sprengen würden.

nisch hoch beanspruchte Teile

- Optimale Ausnutzung der Ressourcen der verfügbaren Gerätegenerationen, d.h. insbesondere Weiternutzung bestehender älterer Rechnersysteme
- Möglichst lange Laufzeiten der Geräte
- Unterstützung einer breiten und unterschiedlichen Hardwarepalette
- Finanzielle Einsparungen durch geeignete Wahl der Software
- Konfiguration für ein ausgedehntes Netzwerk mit der Möglichkeit späterer Erweiterung
- Reduktion des benötigten Aufwands für einen Einzelarbeitsplatz durch Verzicht auf Festplatte und Installationslaufwerke, wie CD-ROM-, DVD- oder Diskettenlaufwerk
- Flexible Software-Ausstattung und einheitliche Präsentation gegenüber den Benutzern
- Einfache Installations- und Wartungsumgebung
- Zentrale Überwachungsinfrastruktur zur schnellen Information bei verschiedenen Ausfallszenarien
- Unkomplizierte Verwaltung und Kennzeichnung der Maschinen für Nutzer und Administratoren

Ein weiterer Aspekt ist die dezentrale Aufstellung der Arbeitsplatzrechner, da der Campus der Universität sehr weiträumig über die Stadt verteilt ist. Trotzdem sollten alle wichtigen administrativen Aufgaben und entscheidenden Server, die zentralen Rechner mit vielfältigen Diensteangeboten zur Steuerung der Arbeitsplatzmaschinen, an wenigen zentralen Stellen gebündelt werden. Voraussetzung hierfür ist die unproblematische Fernwartbarkeit und das einfache Monitoring der Client-Geräte.
Trotz der unterschiedlichen Hard- und Software-Ausstattung sollen alle Maschinen möglichst über ein einheitliches Administrations-Interface zu verwalten sein, um den Aufwand der Installation und des Updates gering zu halten. Ein anderer wichtiger Punkt ist die Perspektive zur leichten Inventur, da bei einer derart hohen Zahl von Einzelteilen ein Überblick zwingend notwendig ist.
Darüberhinaus soll es möglich sein die eingesetzten Rechner in eine DNS-Datenbank zu übernehmen und Konfigurationen für verschiedenste Überwachungsaufgaben zu erstellen.

2.4 Vorgehensweise

Im Rahmen dieser Arbeit kann es verständlicherweise nicht gelingen, das Lösungskonzept zu Aufbau und Administration eines Rechernetzwerkes von ca. 400 Rechnern unterschiedlicher Konfiguration vollständig zu dokumentieren. Die beiden alternativen methodischen Vorgehensweisen - vergleichende Darstellung unterschiedlicher Lösungsmöglichkeiten einerseits, genauere Ausarbeitung eines weitergehenden Konzepts andererseits - müssen daher gegeneinander abgewogen werden.
Der Schwerpunkt der vorliegenden Arbeit basiert darauf, die wesentlichen Schritte zur Entwicklung einer den funktionalen Anforderungen und Möglichkeiten angepaßten Rechnerumgebung nachvollziehbar zu dokumentieren. Die Installations- und Leistungsmerkmale des Lösungskonzepts eines Datenbank gestützten Rechnernetzwerkes[4] stehen daher im Mittelpunkt. Um dem zweiten Gesichtspunkt - der vergleichenden Begründung der gewählten Lösungen - im Rahmen des Machbaren gerecht zu werden, können zwar alternative Lösungskonzepte nicht gesondert ausgewiesen und entwickelt werden, jedoch sollen die Auswahlkriterien offengelegt werden. Hier kann vorab darauf hingewiesen werden, dass es sich bei dem entwickelten Lösungskonzept um eine Plattform handelt, die bewußt für unterschiedliche Software-Architekturen offengehalten wird.

[4] Zur Grundidee siehe Seite 95. Es wird darum gehen eine Datenbank zur Inventarisierung und Beschreibung aller eingesetzen Rechner aufzusetzen und wesentliche Administrationsaufgaben mit ihrer Hilfe zu erledigen.

Kapitel 3

Aufbau der Arbeit

3.1 Überblick

Die Aufgabenstellung erfordert die gesonderte Betrachtung der verschiedenen im vorherigen Kapitel genannten Aspekte. Die Netzwerkhardware wird als gegeben vorausgesetzt, wobei sich bestimmte Anforderungen aus der Aufgabenstellung ergeben. Netzwerke sind inzwischen integraler Bestandteil von Organisationen und stark standardisiert und bedürfen deshalb in den meisten Fällen keiner besonderen Diskussion mehr. Anmerkungen erfolgen an dieser Stelle nur wenn es um bestimmte Anforderungen seitens der agierenden Server bzw. Clients geht.
Im Zentrum dieser Arbeit stehen daher drei Aspekte: Die Ausarbeitung eines Fundaments für den Benutzer-Desktop, die Verwaltung der Rechnerhardware sowie die hierfür notwendigen zentralen Software-Lösungen. Angesichts sehr heterogener und sich dynamisch entwickelnder Applikationen für die Endnutzer kann es hinsichtlich der Desktop-Lösungen nicht darum gehen, dezidierte Einzellösungen zu entwerfen. Vielmehr steht die Entwicklung einer soliden Grundlage im Mittelpunkt, auf der eine breite Palette unterschiedlicher Anwendungssoftware eingesetzt werden kann. Das entwickelte Konzept läßt sich in zwei Schritte zerlegen: Zunächst gilt es, die Client-Server-Architektur der favorisierten Lösung Thin-Clients (Teil II) auszuarbeiten, ehe in einem zweiten Schritt die Möglichkeiten einer Datenbank gestützten Administration des Rechnerpools dargestellt werden (Teil III). Der letzte Teil widmet sich dem Fazit und dem Ausblick mit den möglichen Erweiterungen. Bevor es in die Erörterung der eben angesprochenen Fragestellungen geht, gibt Kapitel 4 Erläuterungen zur Benutzung und einige Erklärungen wichtiger und häufig verwendeter Begriffe und Abkürzungen. Das Kapitel 7 verschafft einen Überblick, welche Themen der Software-

Integration im Rahmen dieser Arbeit vertieft werden können und welche Teile des Projektes aus Platzgründen und Erwägungen der Ausrichtung nur gestreift werden.

3.2 Thin-Clients auf Linuxbasis

Ziel des zweiten Teils der Arbeit ist es, eine Anleitung zu einer Soft- und Hardwarekonfiguration zu geben, mit der sich größere Rechnerpools auf Basis einer Linuxlösung, welche in einem eigenen Unterabschnitt motiviert wird, steuern lassen.
Im ersten Abschnitt wird es deshalb um die Ausstattung der Arbeitsplatzrechner gehen. Klassische Ansätze weisen hierbei einige entscheidende Schwierigkeiten hinsichtlich verschiedener wünschenswerter Anforderungen auf (Kapitel 5). Aus einer Gegenüberstellung der unterschiedlichen Betriebsmodelle als X-Terminal oder Diskless X-Station lassen sich zunächst die jeweiligen Hardware-Anforderungen herausarbeiten und alternative Konfigurationen skizzieren (Kapitel 6). Einige Besonderheiten der DXS (Kapitel 11) werden am Ende des Teil II besprochen. Für die gegebene Problemstellung und unter den bekannten Restriktionen läßt sich die Wahl des Betriebssystems Linux begründen (Kapitel 6).
Die Einrichtung der Thin-Clients als Endgeräte umfaßt drei Aufgabenbereiche, welche zusammengenommen die Basis der Client-Server-Architektur bilden: Zunächst muss die Auswahl und Installation der Server dargestellt werden (Kapitel 8). Die Einrichtung der Clients läßt sich wiederum in hardware- bzw. netzwerkspezifische (Kapitel 9) und softwarespezifische Probleme (Kapitel 10) unterteilen.
Aus den Anforderungen an die Software der Arbeitsplatzmaschinen - der Clients im beschriebenen Rechnernetzwerk - ergeben sich die Aufgaben, die von den Servern mindestens geleistet werden müssen. Hierzu zählt vor allem die serverseitige Konfiguration der benötigten Netzwerkdienste (Kapitel 8.1).
Aus Sicht der Clients ist zunächst die Erstellung von EPROM's bzw. die notwendigen Anpassungen des Rechner-BIOS für das Booten aus dem Netzwerk wesentlich (Kapitel 9.2). Unter den verschiedenen Ansätzen des Bootens der Arbeitsumgebung soll im Besonderen das Software-Tool "Etherboot" zum Erstellen der Startsoftware hervorgehoben werden. Weiterhin soll die Anpassung der SuSE-Linuxdistribution an die besonderen Bedingungen des festplattenlosen Betriebes skizziert und die benötigten Software-Tools vorgestellt werden. Die Protokolle TCP/IP-basierter Netzwerke: BOOTP/ DHCP, TFTP und NFS werden näher vorgestellt und in ihrer Umsetzung für das Betriebssystem Linux beschrieben.

3.3 Das Datenbankmodul

Die Motivation des Aufbaus einer datenbankgestützten Konfigurationsumgebung leitet sich aus den Konfigurationsanforderungen der Clients ab. Da diese Rechner ohne klassisches Festspeichermedium auskommen sollen, fällt die üblicherweise gewählte Ablage der Konfigurationsparameter auf dem Gerät selbst weg. Die Daten müssen über das Netzwerk beschafft werden und sollten hierfür einfach aufbereitet und bereitgestellt werden können. Durch die unterschiedlichen Beschaffungszeiträume der Hardware, die steigende Leistungsfähigkeit einzelner Komponenten und die relativ hohe Anzahl der Kombinations- und Betriebsmöglichkeiten von PC-Hardware erweitert sich die oben gestellte Forderung auf eine automatisierte Erstellung der Konfigurationsdateien (Kapitel 13, Kapitel 14).
Viele der benötigten Konfigurationsparameter werden zur Generierung der Dateien für das Domain Name System (DNS) benötigt. Da in der Datenbank darüber hinaus Informationen zu einzelnen Hardwareteilen gespeichert und verwaltet werden können, bildet die Datenbank das Herzstück der Lösung. Über die Inventarisierungsaufgaben hinaus ergibt sich die Bedeutung und der Nutzen der Datenbank aus ihrer Kombinierbarkeit mit unterschiedlichen Administrationstools. Die zentrale Verwaltung einzelner Rechner, Komponenten oder auch Konfigurationen kann durch eine entsprechende Erweiterung der Datenbank erheblich ausgedehnt und weitgehend automatisiert werden.
Die Kombination der Datenbank mit Administrationstools bildet daher die Basis der im folgenden entwickelten Betriebsumgebung für größere Rechnerpools. Ergänzung, Fortentwicklung oder Austausch einzelner Hardwarekomponenten, Rechnerkonfigurationen oder Netzwerk-Features wie z.B. der Systemüberwachung oder der DNS-Tabellen stehen auf diese Weise aktuell und synchron zur Verfügung.
Die geeignete Wahl der Datenbank (Kapitel 13) erlaubt vielfältige Varianten des Zugriffs und der Manipulation. Dieses umfaßt webbasierte Steuerelemente über eine Skriptsprache wie PHP, kommandozeilenorientierte Datenbank und Sicherungswerkzeuge, sowie grafische Benutzerschnittstellen auf Basis der beliebten GUI-Bibliotheken[1] QT oder GTK (Kapitel 15). Ein Verwenden der Datenfelder in anderen Datenbanken wird bei der Wahl eines geeigneten Standards erleichtert. Die Zahl der verfügbaren Programmierschnittstellen bestimmt über den Komfort und die Nutzbarkeit der Datensätze.

[1] Graphical User Interface

3.4 Verschiedene Administrationstools

Administrationstools (Kapitel 16) können ausgehend von einer funktionierenden Datenbanklösung in unterschiedlichen Richtungen zum Einsatz kommen bzw. auch erst entwickelt werden. Durch die datenbankgestützte Erstellung der Konfiguration der Diskless X-Stations und X-Terminals kann ein erheblicher Teil des Administrationsaufwandes eingespart werden. Diese Lösung läßt sich auch auf die Einrichtung der meisten Server und weitere angeschlossene festplattenbasierten Systeme durch ein automatisches Installationstool ausdehnen. Auf Basis der Arbeitsumgebung der Diskless X-Stations kann eine einfache Wartungs- und Installationsplattform betrieben werden.
Neben der automatisierten Erstellung von Konfigurationsdateien und der Lösung von Inventarisierungsaufgaben (Kapitel 18) eignet sich die Datenbanklösung auch für weitergehende Kontroll- und Qualitätssicherungsprobleme. Hierzu zählt beispielsweise die aktuelle Überwachung der Betriebszustände (Kapitel 17) der Komponenten von Arbeitsplatzrechnern und Servern, oder der Einsatz unterschiedlicher Skripten für vielfältige Routineaufgaben, der durch Datenbanksteuerung vereinfacht wird. Aber auch der technischen Handhabung angelagerte Bereiche organisatorischer oder betriebswirtschaftlicher Art können von dem Konzept unmittelbar profitieren.

3.5 Möglichkeiten zur Verallgemeinerung

Am Ende dieser Arbeit wird es darum gehen, die Grenzen der vorgestellten Lösung zu skizzieren und ihre Übertragbarkeit auf Bereiche außerhalb der für eine Universität beschriebenen Aufgabenfelder zu beleuchten (Kapitel 19). Das bezieht die Verwendung für den gemeinsamen Einsatz mit anderen Software-Plattformen ein. Weiterhin geht es um die Abschätzung der Einsparpotenziale der vorgestellten Architektur. Dabei geht es auch um die Einordnung von Thin-Clients in das jeweilig gegebene Gesamtkonzept.
Die Bedeutung und Verwendbarkeit von offenen Schnittstellen und Open-Source-Software wird beleuchtet. Darüberhinaus werden Ausblicke eröffnet welche weiteren Schnittstellen und Verbindungen zu anderer Software vorstellbar sind und welche Administrations- und Verwaltungstools entwickelt werden könnten (Kapitel 20).
Die Felder der Datenbank werden auf ihre Brauchbarkeit für die gegebenen Aufgaben untersucht und es werden Erweiterungsvorschläge für denkbare Lösungen entwickelt. Der Gebrauchsnutzen der Datenbank läßt sich weiter erhöhen indem Zusammenhänge, die sich aus bestehenden Informationen ableiten lassen, auf Plausibilität untersucht werden. Weiterhin sollen an die-

ser Stelle Exportfilter zur Aufbereitung der Datensätze für andere Bereiche vorgestellt werden.
Zum Schluß erfolgen Betrachtungen zu Überwachungstools und die über die beispielhaft gezeigte Applikation hinausgehende Anwendungen. Auch hier gibt es eine starke Abhängigkeit von der jeweiligen Aufgabenstruktur und den Möglichkeiten einzelner Überwachungsprogramme und -protokolle zu beachten. Weiterhin werden die Implikationen zum Thema der Systemsicherheit kurz angesprochen.

Kapitel 4

Erläuterungen zur Benutzung

4.1 Bezeichnung von Dateien und Verzeichnissen

Zur Erhöhung der Lesbarkeit werden alle ausführbaren Dateien und Systembefehle durch **Fettdruck**, z.B. **dhcpd** hervorgehoben. Alle Konfigurationsdateien werden *kursiv* gesetzt, um sie aus dem Druckbild herauszusetzen. Da sich die Beschreibungen zum einen auf das Filesystem der Thin-Clients auf dem Server beziehen, aber auch aus Sicht der Clients dargestellt sind, wird immer der absolute Pfad angegeben. Damit ist immer ersichtlich, wo sich die Datei aus Serversicht befinden. Der Serverteil des Pfades befindet sich dabei in einfachen Klammern, z.B. *(/nfsroot/dxs)/etc/dhclient.conf*. Teilweise werden diese Dateien jedoch erst während des Betriebes des Thin-Clients angelegt (und sind auf dem Server nicht oder nur als leerer Link vorhanden). In den folgenden Kapiteln werden zur Erläuterungen nur Abschnitte der verschiedenen Dateien abgedruckt. Die vollständige Ausgabe befindet sich für alle entscheidenden Dateien im Anhang. Im Anhang werden darüberhinaus genauere Angaben zur Benutzung einiger Tools gegeben. Ausdrucke oder Ausschnitte von Konfigurations- und Programmdateien sind immer in der Schriftart `courier` gesetzt.
Um den Lesefluß nicht stark zu stören, werden viele Erläuterungen als Fußnoten angefügt. Alle Links zu den entsprechenden Webseiten sind jeweils am Fuße der Seite zu finden.
Der Anhang enthält alle wesentlichen Skripten, wie sie für die Beispiellösung erstellt wurden. Handelt es sich um Anpassungen vorhandendener Skripten,

so wurde die Kommentarsprache je nach Vorgabe gewählt. Um den Umfang der Dokumentation nicht zu sprengen, liegt eine CD-Rom bei, welche alle Aspekte des Projektes incl. Beispiel-Filesystem etc. enthält. Die Autoren der verwendeten Skripten sind jeweils im Header genannt.

4.2 Begriffserklärungen

Im folgenden werden einige Begriffe und Abkürzungen erläutert, die im Text häufig verwendet werden.

Client Ein Clientrechner ist im Gegensatz zu Servern der Nutzer von Diensten, die Server anbieten. Fast alle Internetprotokolle basieren auf dem Client-Server-Prinzip.

Diskless X-Station Diskless X-Station meint Workstations auf PC-Basis die mit einem unix(-ähnlichen) Betriebssystem betrieben werden. Das "X" weist auf die Verwendung der am Massachusetts Institute of Technology (MIT) entwickelten grafischen netzwerktransparenten Benutzerschnittstelle als zentrales Merkmal hin. Diese Geräte binden ihr Dateisystem von einem entfernten Fileserver ein, und nicht wie noch meist üblich, von einer lokalen Festplatte (diskless). Sie gestatten dem Benutzer das lokale Ausführen von Applikationen, sowie den Zugriff auf alle Laufwerke, installierte Erweiterungshardware (Audio, Video, SCSI, USB ...) und angeschlossene Peripheriegeräte.

DHCP Dynamic Host Control Protocol. DHCP verwendet UDP und benutzt für den Serverkanal Port 67 und den Clientkanal Port 68.

DXS siehe Diskless X-Station.

GPL GNU General Public License bildet die Grundlage der freien Software-Entwicklung. Sie wurde geschaffen, um die Freiheit selbstgeschriebenen Codes zu garantieren, was dessen Weitergabe, Veränderung, Vervielfältigung etc. anbetrifft. Die genauen Bedingungen dieser Lizenz kann den vielfältigen im WWW verbreiteten Kopien entnommen werden[1]. Ein großer Teil der hier benutzten Software basiert auf dieser oder ähnlichen Lizenzmodellen.

[1] z.B. direkt von http://www.gnu.org/copyleft/gpl.html

Kernel Mit Kernel (engl.) wird der Kern, d.h. das eigentliche Betriebssystem, welches die Interaktion zwischen der Hardware und der Software übernimmt, bezeichnet. Linux ist daher strenggenommen "nur" dieser Kern, alle anderen Programme und Utilities setzen auf diesem Kern auf und stehen üblicherweise nicht nur unter Linux zur Verfügung.

LAN Local Area Network. Dies meint Netzwerke einer geringen bis mittleren Ausbreitung, die sich inzwischen fast ausschließlich der Ethernet-Technologie bedienen. Die Übertragungsgeschwindigkeiten sind mit 10 Mbit (klassisches Ethernet), 100 Mbit (Fast-Ethernet) und inzwischen auch 1 Gbit angegeben. Weitere Technologien bedienen sich ATM (Asynchronous Transfer Mode), TokenRing- oder FDDI-Technologie (Fiber Distributed Data Interface).

Linux Workstation Mit Linux Workstation wird ein komplett ausgestattetes und an bestimmte Aufgaben angepaßtes Gerät für Officeanwendungen, Administrationsaufgaben, technische Entwürfe, Bildbearbeitung etc. bezeichnet. Dieser Gerätetyp unterscheidet sich von einem Server insofern, als das es sich um einen Arbeitsplatzrechner handelt, der mit allen üblichen Eingabe- und Ausgabegeräten ausgestattet ist und nicht vielen Benutzern einen bestimmten Dienst anbietet.

NFS Network File System. NFS ist ein UDP- oder TCP-basiertes Protokoll[2], das Dateisysteme über ein TCP/IP-Netzwerk zur Verfügung stellen kann.

Perl Perl[3] ist eine interpretierte Skriptsprache, die sich im Bereich der Stringverarbeitung durchgesetzt hat. Sie steht unter der GPL (siehe oben) für alle gängigen Unix-Architekturen, aber auch für Mac-OS und Microsoft-Betriebssysteme zur Verfügung. Diese Programmiersprache kann mittels Modulen erweitert werden. Solche stehen inzwischen für jeden Anwendungsfall[4] in weltweit zugänglichen Archiven[5] zur Verfügung.

PHP PHP[6] ist eine für Webanwendungen optimierte Skriptsprache, welche z.B. über ein Modul des Webservers Apache[7] ausgeführt werden kann.

[2] Ab NFS Version 3 steht auch TCP als Übertragungsprotokoll zur Verfügung
[3] Einstieg über [M8] bzw. http://www.perl.org
[4] die Umsetzung von Netzwerkprotokollen oder die Schnittstellen zu bestimmten Anwendungen
[5] Das CPAN
[6] Einstieg über http://www.php.org
[7] www.apache.org

RFC Request for Comment (engl.) heißen die Texte, die beschreiben, was das Internet im Innersten zusammenhält. In diesen Dokumenten werden neue Netzwerk-Protokolle definiert, ihre Änderungen und Erweiterungen beschrieben und zur Diskussion gestellt.

RPM Das RedHat-Packagemanager-Format hat sich im Linuxumfeld inzwischen als Quasi-Standard zur Verteilung und Installation von Software-Paketen durchgesetzt. Eine ganze Reihe von Linuxdistributoren und viele Entwickler verwenden dieses Format. Es existieren eine ganze Reihe von Applikationen zur Steuerung der Software-Auswahl und Installation und Deinstallation.

Server Der Server ist auf der einen Seite Diensteanbieter im klassischen TCP/IP-Client-Server-Modell, d.h. er stellt - meistens zentral - bestimmte Funktionalitäten, wie Mail-, File- und Webdienste oder Applikationen zur Verfügung. Benutzer können sich an einem Server anmelden, werden aber nur in den seltensten Fällen physisch vor dem Gerät sitzen.

Skript meint zumeist ein Stapel (engl. Batch) meist nicht interaktiver Programme, die in einer ausführbaren Datei zusammengefaßt sind. Die Shell, der Unix-Kommandointerpreter, stellt neben anderem eine solche Funktionalität zur Verfügung, weshalb man häufig von Shellskripten spricht. Diese "Programme" bilden die Grundlage der Steuerung eines jeden Unix-Systems. Darüberhinaus gibt es eine ganze Reihe interpretierter Programmiersprachen, welche auch zur Lösung von Aufgaben der Systemadministration eingesetzt werden. Eine häufig verwendete Sprache ist Perl[8]. In dieser Programmiersprache sind die meisten der hier vorgestellten Beispielskripten abgefaßt.

System Management Bus Der System Management Bus (SMB) stellt auf modernen (PC-) Mainboards ein Standard zur Überwachung wichtiger Komponenten dar. Es lassen sich darüber beispielsweise Informationen über Lüfterdrehzahlen und Temperaturen gewinnen.

TFTP Trivial File Transfer Protocol. TFTP stellt eine stark vereinfachte Version des FTP dar und arbeitet auf Basis von UDP Port 69.

UDP User Datagram Protocol. UDP ist Teil von TCP/IP und stellt eine Implementierung der Transportschicht dar. UDP arbeitet verbindungslos.

[8] Siehe Erläuterungen in diesem Abschnitt

X11 bzw. XFree86 X11 bringt die grafische Oberfläche auf den Unix-Desktop und definiert die Netzwerkschnittstellen. Als Protokoll wird XDMCP, das X-Display Managagement Control Protocol verwendent. Der X-Server, bei Linux meistens durch die Implementierung von XFree86 realisiert, leistet die Grafikausgabe auf der lokalen Maschine, muss also die entsprechenden Hardwaretreiber enthalten sowie auf Monitor und Grafikkarte anpaßbar sein.

X-Terminal Meint hier ein festplattenloses System, welches in erster Linie mittels XDMCP (siehe X11 bzw. XFree86) mit einem oder mehreren Servern Kontakt aufnehmen kann. Dabei erfolgt die Ausgabe des Grafikoberfläche des Servers lokal auf der Maschine. Die Benutzereingaben durch Tastatur und Maus werden über XDMCP an den Server weitergereicht.

Teil II

Thin-Clients auf Linuxbasis

Kapitel 5

Arbeitsplätze an Net-PC

5.1 Aufbau des zweiten Teils

Ausgehend von der Problemstellung, ein funktionsfähiges Rechnernetzwerk unter gegebenen Anforderungen und Restriktionen zu erstellen, soll im folgenden zunächst diese näher spezifiziert, alternative Lösungsmöglichkeiten diskutiert und die Auswahl des Lösungsansatzes motiviert werden (Kapitel 6). Nach der Begründung der Thin-Clients gilt es in einem zweiten Schritt dann die Basisprozeduren zur Installation der favorisierten Thin-Clients auf Linuxbasis darzustellen (Kapitel 8, 9, 10).
Das Konzept eines Thin-Clients-Netzwerks muss an dieser Stelle als die Basislösung angesehen werden, für die die Installations- und Konfigurationsprozeduren (die notwendigen Interaktionsschritte zwischen Servern und Client) sehr weitgehend dokumentiert werden können. Der Abschnitt II. dieser Arbeit basiert auf einem längerfristigen Projekt, dessen Einzelaspekte bereits vor einiger Zeit im *Linux-Magazin* vorgestellt wurden[1]. Einige Überlegungen finden sich in ähnlichen Thin-Client-Projekten[2] auf Linuxbasis wieder. Von hier ausgehend sind dann wiederum unterschiedliche Erweiterungsmöglichkeiten denkbar, wie sie insbesondere die Datenbanklösung bietet, die in Teil III vorgestellt wird.

[1] siehe hierzu die beiden Artikel in [Z5] und [Z6]
[2] einen guten Einstieg bieten hier [W1] und [L1]

5.2 Vorbetrachtung Probleme klassischer Lösungen

Viele angebotene klassische Lösungen arbeiten noch immer mit der fast autonomen Arbeitsstation aus der Vor-Netzwerk-Ära. Hier wird jeder Rechner mit allen notwendigen Komponenten zur Installation, festplattenbasiertem Betrieb und in vielen Fällen zum Backup ausgestattet. Dieser Grundidee sind viele der heute verfügbaren PC-Betriebssysteme stark verhaftet. Auf Basis dieser Idee erfolgte vor 20 Jahren der Einstieg in das PC-Zeitalter, welches die Zeiten der großen Mainframe-Architekturen, welche zuvor vom Ende der 60er Jahre an domonierten, ablöste, bzw. diese in Randbereiche zurückdrängte. Dieser Vorgang läutete einen Paradigmenwechsel ein. Das nun angebrochene neue Rechnerzeitalter schuf die Grundlage des gewaltigen Wachstums der letzten Jahre in der Informationstechnologie.
Mit diesem Wachstum gingen jedoch auch etliche Probleme einher, die sich bei der mittlerweile erreichten fast allgegenwärtigen Verbreitung der PC-Systeme stärker bemerkbar machen. Dazu zählt sicherlich die Unterschätzung der Potenziale von Netzwerken und die Konzentration auf den immer besseren Ausbau des Einzelplatzes mit nur schwacher Berücksichtigung seiner Einbindung in übergeordnete Strukturen. Die Betriebssysteme der Personalcomputer wurde lange Zeit auf Einzelplatzbetrieb zugeschnitten, d.h. Besitzer bzw. Anwender des Gerätes war auf diese Weise automatisch Systemadministrator und hatte uneingeschränkten Zugriff auf alle Ressourcen. Mit der Einführung vernetzter Rechner gehen aber zum einen größere Sicherheitsprobleme einher und zum anderen entspricht das Ein-Benutzer-Konzept in vielen Fällen nicht mehr den Anforderungen flexibler Arbeitsplätze.
Von den Mainframelösungen lassen sich jedoch einige Ideen wieder aufnehmen. Wo es zu ihrer Zeit darum ging, rare Compute-Kapazitäten möglichst effizient zu nutzen, ist dieses sicherlich nicht mehr das vordringliche Anliegen. Die zentrale Lösung (meistens wegen der schieren Größe der Maschinen) hatte zumindest aus Sicht der Administratoren große Vorteile. Die Clients, meistens Text-Terminals, waren über serielle Leitungen mit der Hauptmaschine verknüpft und verfügten nur über die notwendige Logik zur Zeichendarstellung auf dem Bildschirm und der Ausgabe auf einen Zeilendrucker. Bei Defekten konnten die Endgeräte einfach ausgetauscht werden, da alle Vorgänge auf dem Mainframe stattfanden.
Prinzipiell gibt es unterschiedliche Ansätze zum Aufbau eines Rechnernetzwerkes. Die favorisierte Linux-Lösung hängt von unterschiedlichen Parametern ab, die teilweise typisch für Universitätsnetzwerke sind, sich teilweise aus einem gewünschten Anforderungskatalog ergeben und letztlich in einem

Administrationskonzept konvergieren, wie es in Teil III entwickelt wird. Aktionsparameter, die für das vorliegende Projekt maßgeblich waren, sind:

- Rechner-Arbeitsplatzanzahl
- Stabilität und Robustheit des verwendeten Systems
- Sicherheitsaspekte
- Kostenreduktion

5.2.1 Steigende PC-Arbeitsplatzanzahl

Ein Anwachsen der Zahl von EDV-gestützten Arbeitsplätzen, wie sie für größere Organisationen - z.B. Unternehmen, Behörden, Universitäten - typisch ist, bringt mit den bisherigen Lösungen der "klassischen" PC-basierten Client-Server-Installationen wachsende Kosten und steigenden Administrationsaufwand mit sich.
Das Studierendennetz an der Universität Göttingen besteht aus über 400 verteilten Rechnerarbeitsplätzen, die teilweise unterschiedlich konfiguriert sind oder sein müssen (z.B. für spezielle Anwendungen, wie Scannen und Grafikbearbeitung, bzw. CD-Erstellung), und zugleich für die Benutzung durch eine Großzahl von Anwendern an wechselnden Orten eingerichtet sein müssen. Dieser Anordnung ist insofern teilweise beispielmäßig, weist aber auch eigene Spezifika auf. Die Anforderungen an das Aussehen der Arbeitsplätze bleiben ähnlich, trotzdem stiege der Administrationsaufwand rasant, würde man auf klassische Lösungen auf der Basis von PC-Workstations[3] zurückgreifen.
Historisch ändert sich mit der Ablösung der alten zentralen Mainframes durch dezentralisierte, später vernetzte PC's, die Struktur der Betriebskosten. Zu den zentralen Servern kommen ein aufwändiges Netzwerk und üppig ausgestattete Arbeitsplatzrechner hinzu. Sind die Einzelsysteme als vollständige Standalone-Lösungen[4] realisiert und ist man nicht bereit, sehr viel Geld in Spezialsoftware zum zentralisierten Management zu stecken (oder diese selbst zu erstellen), gerät man sehr schnell in eine Sackgasse. Das Zeitbudget zur Wartung wird nur noch mit Routineaufgaben gefüllt sein. Es entsteht ein hoher Bedarf an Administration und Ressourcen sowie Probleme mit der Migration in nächsthöhere Rechner- oder Betriebssystemgenerationen.

[3] Meint den handelsüblichen vollausgestatteten PC mit Festplatte, Diskettenlaufwerk, CD-ROM und evtl. Bandsicherungslaufwerk oder CD-Brenner
[4] PC- oder Unix-Workstation

5.2.2 Stabilität und Robustheit der Arbeitsplatzrechner

Weitere Risiken herkömmlicher Architekturen liegen im Defektfall einzelner Komponenten der Arbeitsplatzsysteme. Fällt eine Festplatte oder ein anderer Teil der Hardware aus, sind meistens höhere Aufwendungen nötig, um die Arbeitsstation inklusive ihrer angepaßten Software und Benutzerumgebung wieder in Betrieb zu nehmen. Tücken liegen weiterhin im Zusammenspiel der Hardware[5] und speziellen Anpassungen der User[6].
Eine Möglichkeit ist, die verwendete Hardware an den Rechnerplätzen zu reduzieren, um Ausfallwahrscheinlichkeiten und Belastungen zu reduzieren. Gegenüber vernetzten PC's oder Stand-Alone-Lösungen weisen Diskless X-Stations, aber auch X-Terminals bereits erhebliche Vereinfachungen auf. Die Zahl der mechanisch hoch beanspruchten Teile wird bei Diskless X-Stations auf Wechsellaufwerke für Disketten, CD's, DVD's, etc. beschränkt und sie entfallen bei X-Terminals ganz.
Schließlich kann es durch unbedachtes Ausschalten des Computers bzw. Stromausfälle (die nicht für alle Arbeitsplätze beispielsweise durch unterbrechungsfreie Stromversorgungen kontrollierbar sind) leicht zu Beschädigungen am Dateisystem bzw. zu Totalausfällen von Hardwarekomponenten kommen. Darüberhinaus sind Computeranwender es gewohnt, Rechnerstillstände über den Resetknopf zu beheben, wie es von einem weitverbreiteten Betriebsystem als klassische Lösung nahegelegt wird. Dieses Nutzerverhalten sollte in Betracht gezogen und Systeme entsprechend entwickelt werden.

5.2.3 Sicherheitsaspekte

Sicherheitsaspekte betreffen Backups, Zugriffsrechte auf sensible Daten, sowie den Schutz vor Angriffen aus dem Netz. Herkömmliche PC-Lösungen überlassen Backups - trotz vorhandener Fileserver - weitgehend den Einzelnutzern und sind damit einerseits in hohem Maße zufällig, andererseits kaum kontrollierbar. Demgegenüber verfügen Diskless X-Stations und X-Terminals über kein eigenes Festspeichermedium, sind also bereits so angepaßt, dass alle relevanten Dateien des Betriebssystems, der Anwendungen und Anwenderdaten zentral abgelegt werden müssen. Hierdurch vereinfacht und reduziert sich ein Backup auf wenige zentrale Server, die besser schütz- und kontrollierbar sind. Es muß nun nicht mehr speziell sichergestellt werden, dass ein bestimmter zu sichernder Arbeitsplatz-PC eingeschaltet sein muss und in der kritischen Phase nicht abgeschaltet werden darf.

[5] spezielle Treiber für Boardkomponenten, Laufwerke
[6] persönliche Gestaltung des Desktops, angepaßte Defaults für bestimmte Software

Alle Systemprogramme und Bibliotheken, sowie die meisten Konfigurationsdateien im Rootfilesystem der Thin-Clients werden an den Arbeitsstationen nur ReadOnly zur Verfügung stehen. Eine Manipulation kann daher nur auf Serverseite erfolgen. Die geringere Zahl der Server gegenüber der Gesamtzahl der Arbeitsplätze erlaubt jedoch ein aufwändigeres Sicherheitskonzept ohne den Administrationsaufwand überproportional zu erhöhen. So wird der potenzielle Nachteil einer serverseitigen Sicherheitsfixierung jedoch schnell wieder ausgeglichen.
Weiterhin gibt es Sicherheitsbedenken, wenn an dezentralen Arbeitsstationen sensible Daten liegen oder diese sehr einfach entfernt bzw. kopiert werden können. Server mit sensiblen Daten können besonders gesichert aufgestellt und der Datenzugriff zentral auf bestimmte Bereiche und Zeiten festgelegt werden. Die Vorteile dieser Lösung steigen, je weniger Daten und Zugriffsrechte auf Systemdateien dezentral liegen.
Neue Sicherheitsprobleme entwickeln sich in dem Augenblick, in dem ein rechnergestützter Arbeitsplatz in ein Netzwerk eingefügt wird. Dies betrifft insbesondere auch die hier vorgestellten netzbasierten Lösungen. Einige der benutzten Protokolle sind unabhängig von ihrem Anwendungsgebiet unsicher und angreifbar. Um hier die Sicherheit zu erhöhen, gelten dieselben Regeln wie in einem Netzwerk klassischer Workstations. Das kann z.B. bedeuten, den gesamten IP-Verkehr verschlüsselt ablaufen zu lassen, was jedoch höhere Leistungsanforderungen an die einzelnen Netzstationen stellt. In einer zentralisierten und stark vereinheitlichten Umgebung lassen sich jedoch Sicherheitskonzepte einfacher implementieren, als in einem Netzwerk autonomer Workstations.

5.2.4 Kostenreduktion

Über die Vereinfachung der Administration hinaus, die sich einerseits aus den Verbesserungen o.a. Aspekte (Sicherheit, Stabilität, Wartungsempfindlichkeit etc.) ergeben, bietet eine gut konfigurierte Netzwerklösung weitergehende Perspektiven zu einer optimalen Nutzung vorhandener Ressourcen:

- Hardwarekomponenten - zuerst die Festplatte und das klassische Installationslaufwerk, wie CD-ROM- oder DVD-Laufwerk o.ä. - entfallen und Backup-Systeme sind nur an einer bzw. wenigen zentralen Stellen erforderlich. Die Geräte werden kleiner und platzsparender.

- Mitarbeiter können von der Konfiguration ihrer Rechner, dem Backup-Geschehen und dem Umlernen auf neue Desktop-Umgebungen bei einem Wechsel des Arbeitsplatzes entlastet werden. Die Geräte müssen nicht für ein Backup eingeschaltet sein.

- Die Wahrscheinlichkeit von Ausfällen sinkt durch Reduktion mechanischer und empfindlicher Komponenten. Der Energieverbrauch verringert sich und reduziert dadurch die laufenden Kosten. Die verminderte Lärmabstrahlung erhöht die Ergonomie der Arbeitsplätze.

- Die Anschaffungs- und Updatekosten der Software sollen nicht ins Gewicht fallen.

Ein Netzwerk (Local Area Network) ist in den meisten Fällen schon vorhanden und fällt als zusätzlicher Kostenaspekt nicht ins Gewicht. Ein bestehendes 10Mbit-Netzwerk genügt den üblichen Ansprüchen für den Betrieb von X-Terminals. Die Mehrkosten für den evtl. Ausbau des Netzwerkes auf 100Mbit fallen wegen der gesunkenen Anschaffungskosten für die aktiven Komponenten inzwischen gleichfalls moderat aus. So werden die bei den Clients eingesparten Mittel nicht durch Netzwerkmehrinvestitionen überkompensiert.

Kapitel 6

Wahl des OS und Aufwandsfragen

6.1 Vorüberlegungen

Die Kosten und der Aufwand des Betriebes großer Rechnernetzwerke setzen sich aus mehreren Komponenten zusammen. Zu nennen sind die zu tätigenden Anschaffungen in Form von Hardware und Software-Lizenzen. Deren Benutzung und Wartung erfordert einen stetigen Personal- und Administrationsaufwand. Einsparungen auf der einen Seite lassen sich in gewissem Umfang durch Mehraufwand auf der anderen Seite kompensieren[1]. Wie dieses Verhältnis im konkreten Fall genau aussieht, hängt von den jeweiligen Anforderungen ab.
Im folgenden wird deshalb zuerst der Aufbau des Netzwerkes und anschließend die Wahl des geeigneten Operating Systems[2], sowohl unter Aspekten der Erstanschaffung und laufender Updates, als auch unter dem Gesichtspunkt der Wartbarkeit, der Skalierbarkeit und Kompatibilität zu Hardware und anderer Software motiviert.
Nachfolgend wird überlegt, welche Konfiguration sich für welchen Einsatzzweck eignet. Ein Anhaltspunkt bietet die Zahl der zu betreibenden Geräte oder später noch hinzukommenden Maschinen. Eine schon bestehende Ausstattung, so sie denn weitergenutzt werden soll, wird ebenso in die Betrachtungen einfließen.
Es besteht in Abhängigkeit von der Leistungsfähigkeit der Hardware und

[1] Preiswertere Rechner (mit geringeren Garantiezeiten) erfordern meistens höheren Wartungsaufwand, Markengeräte üblicherweise höhere Anschaffungskosten.
[2] Abk. OS - Betriebssystem

den Anforderungen der Software die Wahl zwischen verschiedenen Betriebsmodellen: X-Terminal oder Diskless X-Stations.

6.2 Aufbau des Netzwerkes

Im vorgestellten Fall handelt es sich um ein campusweites größtenteils geswitchtes[3] LAN. Hier werden einzelne Räume und Gebäude mittels Gigabit- bzw. Fast-Ethernet untereinander verbunden. Im zentralen Gebäudekomplex befindet sich der Serverraum und ein Großteil der Arbeitsplatzrechner. Viele weitere Räume und Standorte sind jedoch auch weiter entfernt, eine Situation die der in ähnlich großen Unternehmen, Organisationen oder Behörden entsprechen dürfte.
Solche Netzwerke sind in den meisten Fällen inzwischen Standard oder werden zumindest bei weiteren Ausbaustufen angestrebt, da sie unabhängig von der je vorgestellten Lösung zur unternehmens- bzw. organisationsinternen Kommunikation benötigt werden.
Ziel des Projekts war zunächst, alle administrativ kritischen Server zentral an einem Ort zu bündeln, um in heiklen Situationen schnell reagieren zu können, und möglichst wenig Reibungsverlust bei Koordinationsproblemen zu verursachen. Insbesondere unter Sicherheitsaspekten in den verschiedenen Hinsichten[4] bietet Zentralisierung optimale Gestaltungs- und Kontrollmöglichkeiten.
Darüberhinaus erlaubt die beschriebene Gestaltung des Netzwerkes die Bündelung Know-How-intensiver Tätigkeiten an wenigen (an einem einzigen) zentralen Ort(en), womit Wege- und Kommunikationsverluste vermieden werden. Die Wartung der Arbeitsplätze beschränkt sich auf einfachere Tätigkeiten, die auch von IT-fremden Personen ausgeführt werden können, wie das Aufstellen, Anschließen und Austauschen von Maschinen.

6.3 Wahl des OS: Linux

Linux ist ein Multiuser-Betriebssystem mit Multisession-Funktionalität. Es leitet sich von den klassischen Unixen, welche seit Jahren im Server- und Workstationmarkt etabliert sind, ab und blickt auf eine inzwischen 10-jährige Entwicklungsgeschichte zurück. Es verbindet die ausgereifte Software-Architektur erprobter Betriebssysteme mit der Hardware eines dynamischen Massenmarktes. Alle modernen Software-Entwicklungen, wie z.B. Journa-

[3]"Switching" ist eine Technologie zur Verbesserung des Datendurchsatzes in Netzwerken

[4]siehe hierzu 5.2.3

ling Filesysteme, Logical Volume Management, Softraid etc. werden von Linux unterstützt. Inzwischen steht dieses Betriebssystem in der Portierung auf weitere Hardwareplattformen an der Spitze.
Es gibt mittlerweile eine Reihe von Herstellern von X-Terminals und Thin-Clients mit Multi-Connect-Funktion. Diese Geräte können auf ein oder mehrere Servertypen zugreifen. Es gibt Lösungen, die nur das Windows-Metaframe-Format (bzw. Terminalserver) bzw. XDMCP unterstützen, aber auch Hybridlösungen. Die Geräte klassischer Terminalhersteller sind verhältnismässig teuer und proprietär, d.h. man legt sich, wie z.B. mit der Sun-Ray-Lösung[5] von Sun Microsystems, auf ein Konzept fest.
Viele Dritthersteller bieten Hybridlösungen an, die mehrere Protokolle unterstützen. Diese Geräte basieren aber in den häufigsten Fällen bereits auf Linux, das meistens auf einer "Solid State Disk" untergebracht ist. Lösungen auf WindowsCE-Basis zeichnen sich bei gleicher Funktionalität durch einen höheren Preis aus. Vor- und Nachteile der erstgenannten Lösung liegen in einer gewissen Autonomie des Geräts aber im höheren Aufwand die Software zu updaten und höheren Kosten für das Speichermedium. Die WindowsCE-Lösung bietet die üblichen Vor- und Nachteile kommerzieller Betriebsysteme: Eine gute Anbindung an die Produkte des gleichen Herstellers, meistens aber die Ignoranz gegenüber alternativen Lösungen und hohe Kosten bei Updates.
Obwohl Linux nicht den höchsten Marktanteil als Betriebssystem am PC- und Servermarkt hat, weist es einige hier bemerkenswerte Vorteile auf: Für dieses freie OS[6] fallen keinerlei Lizenzkosten an, die Installationsdatenträger müssen nur ein einziges Mal beschafft werden bzw. sind aus dem Internet zu kopieren. Die Einbindung des Serverfilesystems auf eine Reihe von Clients zur Reduktion des Administrationsaufwandes und notwendigen Festplattenplatzes verletzt dadurch keine Rechte Dritter.
Die entscheidenden Vorteile des Einsatzes von Linux als Client-Betriebssystem ergeben sich aber neben der unmittelbaren Ersparnis von Lizenzierungskosten durch die gegebenen Möglichkeiten von Netzwerk- und Hardwareunterstützung, die Offenheit für unterschiedliche Anwendungen und die Perspektiven einer differenzierten, aber dennoch zentralisierten Administration.
Das Betriebssystem Linux ist umfassend dokumentiert: Zum einen erlauben die offengelegten Quellen ein direktes "Lesen" seiner Funktionsweisen und -mechanismen, zum anderen hat das inzwischen große Interesse der kommerziellen und nichtkommerziellen Anwender eine unüberschaubare Fülle an Literatur zum Thema hervorgebracht, welche eine gute Einarbeitung in

[5] siehe [Z3], Seite 66
[6] Operating System (engl.) Betriebssystem

die Thematik ermöglichen. Zu nennen wären an dieser Stelle sicherlich die erste deutschsprachige Darstellung des "Linux-Anwender-Handbuches"[7], der umfassende Überblick von Kofler[8] oder auch der gute Einstieg von Hantelmann[9].

6.3.1 Netzwerkunterstützung

Linux ist wie andere Unixe auch ein Netzwerkbetriebssystem, d.h. alle entscheidenden Fähigkeiten zur Kommunikation mit anderen Rechnern sind bereits in umfangreicher Form vorhanden. Der Betriebssystemkern unterstützt von sich aus Dateisysteme, die ausschließlich über ein Netzwerk agieren und auf Festplatten verzichten können sowie die automatische Netzwerkkonfiguration auf der Ebene des Internet Protokolls. Damit läßt sich ein festplattenloser Betrieb als X-Terminal und Diskless X-Station komplikationslos realisieren. Hierin liegt eine der Kernqualitäten des Systems.

6.3.2 Hardwareunterstützung

Inzwischen gibt es Linuxkernel für eine Vielfalt von Prozessorarchitekturen. Die gängige PC-Hardware, um die es hier gehen wird, unterstützt Linux fast ausnahmslos. Durch die dynamische Entwicklung ist bei Erscheinen neuer Hardware recht schnell mit einer Unterstützung durch dieses Betriebsystem zu rechnen. Dieses gilt genauso für zügige Bugfixes und Verbesserungen. Linux ist sehr flexibel. Da die Voraussetzungen für einen plattenlosen Betriebsmodus schon eingebaut sind und das System selbst ressourcenangepaßt eingerichtet werden kann, gelingt es, auch ältere Rechnergenerationen wie 486er und 586er noch als X-Terminal einzusetzen. Sowohl Kernel als auch die Software-Auswahl lassen sich sehr hardwarenah optimieren.
Im vorliegenden Fall ließen sich durch diese Möglichkeit zur Weiterverwendung der bestehenden Ressourcen für die Universität erhebliche Beschaffungskosten einsparen. Auch kann die Innovation neuer Rechnergenerationen auf diese Weise in die vorhandenen Potenziale eingepaßt werden.

6.3.3 Applikationen und Offenheit von Linux

Es existieren mehrere sogenannte Linuxdistributionen, Zusammenstellungen von Basissoftware und Erweiterungspaketen, die aufeinander abgestimmt

[7] [M6] der Literaturliste war das erste deutschsprachige Linuxhandbuch, welches auf den Markt kam.

[8] [M7] liefert eine ausführliche Darstellung von Installation, Konfiguration und Anwendung von Linux

[9] [M5] gibt einen guten Überblick zur Benutzung der Standardkommandos und zur Shell-Programmierung

sind. Fast alle wichtige Standardsoftware, wie Officeanwendungen oder betriebswirtschaftliche Programme stehen inzwischen auch unter Linux zur Verfügung. Rechnet man die Programme hinzu, die über einen Emulator zur Verfügung stehen, wird man kaum einen Bereich finden, der sich nicht auch über Linux abdecken läßt. Damit läßt es sich fast für die gesamte Bandbreite an üblicherweise auftretenden Aufgaben einsetzen, was den Administrationsaufwand senkt, da die Betriebsumgebung nicht zu heterogen ist.
Linux kann sehr flexibel an bestimmte Aufgaben anpaßt werden und viele Module können weggelassen, bzw. hinzugefügt werden. So läßt es sich zum einen für eine leistungsarme Umgebung im Betrieb als X-Terminal für die reine Grafikausgabe und Benutzereingabe konfigurieren oder zum anderen ein reiner Router ohne Grafikschnittstelle erstellen. Für den Serverbetrieb lassen sich die Komponenten der Grafikausgabe reduzieren und das System auf IO-Performance optimieren.
Linux steht inzwischen für eine ganze Reihe verschiedener Prozessor- und Hardwarearchitekturen zur Verfügung. Möchte man in einer heterogenen Hardwareausstattung eine homogene Software-Umgebung nutzen, bietet sich Linux als erste Wahl an. Dies gilt auch für plattformübergreifende Software-Entwicklung.

6.3.4 Administration von Linux

Durch die Ähnlichkeit zu den klassischen Unixes der großen Hardwarehersteller und Software-Firmen sowie die Verwendung der seit Jahren weiterentwickelten GNU-Utilities[10] liegen viele Lösungen klassischer Administrationsaufgaben bereits vor. Administrationsmöglichkeiten unter Linux, die darüber hinausgehen, sind ein Thema dieser Arbeit. Die wesentlichen Vorteile sollten im folgenden vor allem im III. Teil deutlich werden. Für die Vorabauswahl spricht jedoch, dass Linux einerseits hochflexible Lösungen zu realisieren erlaubt, für die andererseits meist zugleich kostengünstige bzw. kostenlose Tools schon bereit stehen.
Die durchgängige Berücksichtigung des Netzwerkbetriebes von Linuxmaschinen erlaubt für fast alle Aufgaben eine bequeme Fernadministration. Hierzu steht die hervorragende Kommandozeile (es gibt verschieden mächtige Unix-Shells) zur Verfügung, die auch über schmalbandige Verbindungen ein effektives Arbeiten erlaubt. Darüberhinaus bereitet es keine Schwierig-

[10] GNU is not Unix: Viele Systemadministratoren entwickelten zur Vereinfachung von Wartung uns Systempflege eigene Werkzeuge, die die bestehenden mitgelieferten Tools ersetzten oder in ihrem Funktionsumfang stark erweiterten. Diese Werkzeuge wurden in ihrem Quellcode der Allgemeinheit zur Verfügung gestellt, wodurch eine ständige evolutionäre Weiterentwicklung und Anpassung erfolgte.

keiten grafisch orientierte Anwendungen netzwerktransparent zu starten. Alle wichtigen Laufzeitinformationen, Daten zur eingesetzten Hardware sind dank der "proc"-Schnittstelle des Kernels leicht zu erhalten. Das Autoprobing[11] des Kernels beim Starten bzw. beim Laden der verschiedenen Module erfolgt unter der Ausgabe einer ganzen Reihe nützlicher Angaben. So kann der Administrator bereits aus der Ferne, ohne eine bestimmte Hardware direkt gesehen zu haben, diese bereits einschätzen und die entsprechenden Module des Kernels erstellen bzw. Software einspielen.
Für das Standardkonzept einer Client-Server-Architektur empfiehlt sich die Reduktion auf eine minimale Software-Ausstattung bei X-Terminals bzw. die einfache Einbindung von DXS in das Serverfilesystem und eine Verwendung von PC-Standardkomponenten. Linuxumgebungen können sehr flexibel konfiguriert werden. So kann der Overhead bei der Erstellung des Kernels und der installierten Software durch die Reduktion auf das Notwendige reduziert werden. Die Modularisierbarkeit erlaubt eine Anpassung an die jeweiligen Ansprüche. Die Transparenz der Software hilft dabei unnötige Komponenten für einzelne Einsatzgebiete wegzulassen. Das wird bei der Erstellung der Arbeitsumgebungen der Thin-Clients ausgenutzt. So umfaßt das Filesystem der X-Terminals nur die Applikationen, die zum Start des X-Servers und der Fernadministration und Maschinenüberwachung notwendig sind. Es reduziert sich hierdurch auf ca. 100 MByte. Die Thin-Clients übernehmen die Hauptteile des Filesystems des Servers, welches ca. 2 - 8 GByte umfaßt, in Abhängigkeit der gewünschten Applikationen.

6.4 Welche Konfiguration?

Durch die Möglichkeiten der serverbasierten Administration sinkt der Arbeitsaufwand pro installiertem System sehr schnell bei steigender Arbeitsplatzzahl. Die Server sind nach der Zahl der Arbeitsplätze und der eingesetzten Software zu dimensionieren. Formeln hierfür lassen sich nicht ohne weiteres angeben, da sie von etlichen Variablen abhängen. Jedoch liefern die folgenden Aussagen einige Anhaltspunkte.
Die Server-/Clientkosten fallen jedoch im Gesamtaufwand immer preiswerter aus als die Summe für die Investition in autonome Workstations. Der (Linux-)Server kann unter Umständen entfallen, wenn das OS auf einer Minidisk untergebracht wird. In diesem Augenblick resultiert jedoch ein höherer Update-Aufwand und es ergeben sich starke Einschränkungen durch geringen verfügbaren Speicherplatz auf solchen Medien.

[11] Das automatische "Suchen" nach Geräten, deren Treiberunterstützung im Kernel vorliegt

Der Server muss über eine leistungsstarke NFS-Implementierung und ausreichend Systemspeicher verfügen, um zügige Antwortzeiten des Filesystems zu gewährleisten. Ist eine höhere Konzentration von Clients auf einen Server geplant, kann sich die Investition in einen Gigabit-Backbone lohnen. Aus bisherigen Erfahrungen kann abgeleitet werden, dass eine Zahl von 100 Clients einen Pool von drei bis vier Dual-Prozessormaschinen der mittleren Leistungsklasse[12] nicht zu stark überlastet.
Unterschiede in der Auslegung der Server ergeben sich aus dem Einsatz der entsprechenden Thin-Clients. DXS greifen nur auf das Serverfilesystem zurück, dies aber in großem Umfang. X-Terminals nehmen nur ein Filesystem von ca. 50-150 MByte Festplatten Speicher auf dem NFS-Server in Anspruch, alle Userprozesse werden jedoch auf den Applikationsservern ausgeführt.
Leider liegen noch keine genauen Statistiken über das Userverhalten vor, so dass exakte Aussagen an dieser Stelle nicht möglich sind. Es gibt aber starke Abhängigkeiten von der für die Benutzer angebotenen Software-Ausstattung[13].
In der beschriebenen Umgebung aus 100 Clients und drei (bis vier) Servern kann auch bei einem Ausfall oder Update von einem Gerät problemlos weitergearbeitet werden. Es ist sinnvoll, die Server redundant auszulegen, da dann der Ausfall eines Gerätes das Weiterarbeiten an den Arbeitsplätzen nicht stört. Updates der Servermaschinen lassen sich leichter realisieren, weil sich ein System aus dem Nutzerbetrieb nehmen läßt.
Durch die Verlagerung der Nutzerprozesse auf die DXS ist es nicht mehr erforderlich, dass einzelne Benutzer auf dem Server arbeiten. Hierdurch läßt sich die (Ausfall-)Sicherheit erhöhen und unbedachte oder bewußte Denial of Service Attacken[14] werden vermieden.
Die gängige Hardware wird von Linux inzwischen problemlos unterstützt und es gibt keine Schwierigkeiten mit handelsüblichen Netzwerkkarten. Alle benötigten Software-Tools stehen kostenlos (unter der GPL) zur Verfügung.

6.5 Einordnung von Thin-Clients

In diesem Abschnitt wird es um die Einordnung des festplattenlosen Rechnertyps gehen, welcher im vorgestellten Lösungsansatz verwendet wird. Eine sehr gute Überblicksdarstellung läßt sich in der Fachzeitschrift *iX* finden, die

[12] Derzeit: Dual-Celeron, 500 Mhz bei 66 Mhz Bustakt und 512-1024 MByte RAM

[13] Gerade die neue Version der graphischen Oberfläche KDE, das Officepaket StarOffice, Netscape Version 6 und Java-basierte Tools erhöhen die Hardware-Anforderungen.

[14] Häufig auch als DoS abgekürzt, meint Denial of Service Angriffe auf ein System, welches dieses so stark belastet, dass von einem regulären Betrieb nicht mehr gesprochen werden kann. Dieser Zustand kann z.B. eintreten, wenn ein Benutzer sehr viele Instanzen eines aufwändigen Programmes startet, sehr viele Netzwerkverbindungen initiiert etc.

ein Heft stark auf diese Thematik[15] ausgerichtet hat. Nach der Begriffsverortung erfolgt die Aufwandsabschätzung der einzusetzenden Hardware und Software.
Aus der gewählten Software- und Hardwarekonfiguration ergeben sich bestimmte Anforderungen an die eingesetzten Server. Es wird deren Einrichtung und die Einrichtung der Clientmaschinen beschrieben. Dabei werden die zugrundeliegenden Netzwerkprotokolle und Programme in ihrem Zusammenspiel näher erläutert.

6.5.1 Begriffsbestimmung

Unter dem Begriff "Thin-Client" werden grafische PC-Arbeitsplätze oder klassisch Terminals[16] subsummiert, die sich in erster Linie durch Reduktion von PC-Workstations unterscheiden und über keine Festplatte oder nur sehr eingeschränkten Festspeicher verfügen. Thin-Clients sind die Antwort auf die steigende Anzahl vernetzter PC-Arbeitsplätze sowie ausufernden Ressourcen- und Administrationsaufwand.
Thin-Clients oder vor einigen Jahren noch Network Computer (NC) bzw. Net-PC spielen im Marktgeschehen wieder eine Rolle. Die Abgrenzung seitens der Hersteller ist hier nicht sehr eindeutig, was aber hier nicht von Bedeutung ist, da die hier vorgestellte Lösung auf klassischen PC-Komponenten basiert, welche durch ihre Präsenz am Massenmarkt die kostengünstigsten sind. Viele kleinere Anbieter von Thin-Clients gingen auch deshalb dazu über, ihre Systeme auf hochintegrierter und platzoptimierter PC-Hardware aufzusetzen.

6.5.2 Bedeutung von Thin-Clients

Der Gerätetyp des Thin-Clients erlaubt es die Administration zu zentralisieren, die Verwaltung der Ressourcen zu vereinfachen und die Kosten des Einzelarbeitsplatzes zu senken. Thin-Clients sind wesentlicher Bestandteil eines Maßnahmenbündels zur Kostensenkung des Rechnerbetriebes. Als reine Terminallösung, dem "server based computing", genügt eine geringe Rechenleistung, um den Anforderungen der Grafikausgabe und Benutzereingabe gerecht zu werden. Linux ist das ideale Betriebssystem für solche Einsätze, da zum einen keine Lizenzkosten anfallen und damit die Zahl der

[15] siehe [Z3]: Hier werden einige Ansätze von Thin-Clients sowohl aus Hardware- als auch Softwaresicht vorgestellt, verglichen und bewertet.

[16] Der Begriff "Terminal" wurde in der Zeit der Mainframearchitekturen für die zumeist an serielle Leitungen angeschlossenen Benutzerarbeitsplätze verwendet. Diese Arbeitsstationen arbeiteten zu dieser Zeit üblicherweise im Textmodus (25 Zeilen und 80 Spalten Bildschirmauflösung)

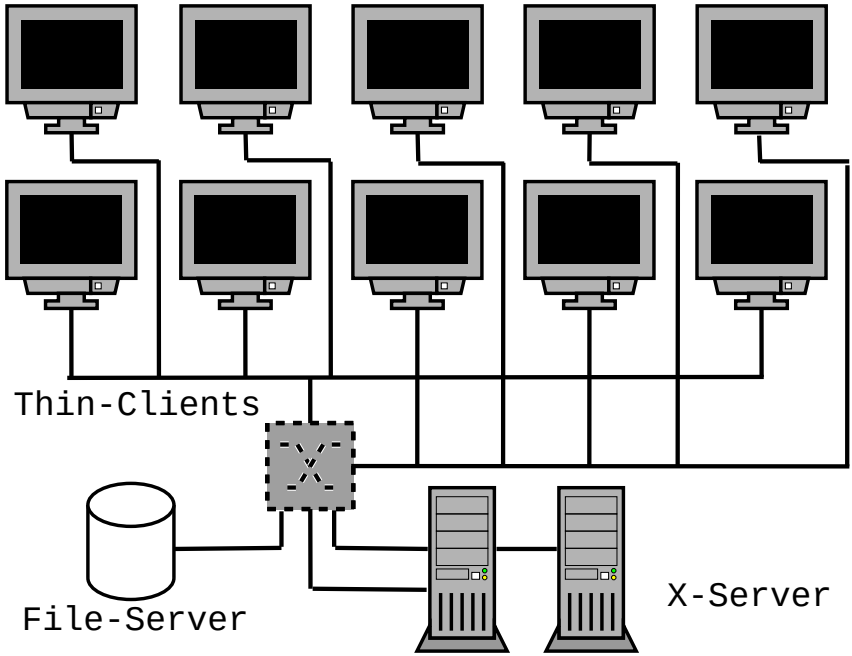


Abbildung 6.1: Typische Client-Server-Konstellation

Geräte ohne Probleme skalierbar ist. Zum anderen bringt es alle wichtigen Fähigkeiten zum plattenlosen Betrieb im Netzwerk bereits mit.
Thin-Clients können unter Linux in zwei Betriebsmodi verwendet werden: Als reines X-Terminal, bei dem alle Applikationen auf dem Server laufen oder als festspeicherlose Workstation auf der die Userprozesse lokal ausgeführt werden. Die Hardware-Anforderungen der X-Terminallösung sind sehr gering (486DX66 mit 16-32 MByte und guter Grafikkarte/Monitor), die Anforderungen an Diskless X-Stations werden durch die eingesetzte Software[17] bestimmt.

[17] Ein einfacher Internetarbeitsplatz mit Browser, Mailprogramm und einfachen Officeanwendungen kommt mit einer CPU ab 266 Mhz und 64-128 MByte Arbeitsspeicher aus. Für grafiklastige Anwendungen, Bildbearbeitung und Software-Entwicklung sollten Prozessoren ab 300 Mhz und 128 MByte RAM zum Einsatz gebracht werden.

6.5.3 X-Terminal oder Diskless X-Station?

X-Terminals stellen ausschließlich die Grafikoberfläche des Servers dar, wobei sie die Möglichkeiten des XDMC-Protokolls der Netzwerktransparenz ausnutzen. Sie nehmen die Benutzereingaben[18] zur Steuerung der Userprozesse auf dem Server entgegen. Ihre Netzwerk- und Hardwareansprüche sind gemessen an heutigen Standards eher gering. Jedoch steigt die Serverlast mit der Zahl der Benutzer und der Zahl ihrer offenen Applikationen. Sie kann auf der Serverseite dann ein Maß erreichen, das flüssiges Arbeiten verhindert.
Die Ausgabe von Audiosignalen ist über das NAS[19] möglich. Der Anschluss von Druckern läßt sich relativ problemlos realisieren. Die Einbindung von Peripheriegeräten und der Zugriff auf Wechselmedien ist jedoch mit höheren Schwierigkeiten verbunden.
Wenn die Leistung aktueller Hardware zum gleichen Preis steigt, wachsen die Anforderungen und Erwartungen seitens der Benutzer und der Software mit. Hinzu kommen neue Nutzungsformen des klassischen Rechners und des Internets; die Zahl der an einen PC anschließbaren Peripheriegeräte werden permanent ausgeweitet, weshalb man in vielen Fällen gleich auf Diskless X-Stations (DXS) setzen wird.
Diese sind vollwertige Arbeitsstationen, die ausschliesslich ihr Filesystem und ihre Konfiguration vom Server beziehen. Damit entlasten sie den Server von Userprozessen und stellen dem User an seiner Maschine ein definiertes Leistungspotential zur Verfügung. Diskless X-Stations erlauben die Einrichtung von Multimediaarbeitsplätzen, den Zugriff auf Wechselmedien und den Anschluss etlicher Peripheriegeräte, wie Scanner, Kameras, ZIP-Laufwerke etc. Die Leistungsanforderungen dieser Maschinenklasse orientiert sich an den Userbedürfnissen sowie der eingesetzten Software und wird mit diesen mitwachsen (müssen). Mögliche Engpässe sind daher höchstens auf Clientseite zu erwarten.
Die Ähnlichkeit der Implementierung von X-Terminals und Diskless X-Stations stellt hochflexible Umgebungen sicher und ermöglicht die einfache Migration des Thin-Clients zwischen beiden Betriebsmodi: Bei steigenden Anforderungen ergibt ein Umstieg auf den Typ der Diskless X-Station Sinn, sinkende Leistung der eingesetzten Hardware im Verhältnis zur aktuellen legt eine Umwandlung in ein X-Terminal nahe.

[18] üblicherweise per Maus und Tastatur
[19] Network Audio System

6.5.4 Einsatz als Embedded System und im Clusterbetrieb

Das Einsatzfeld von Thin-Clients ist wesentlich größer und nicht nur auf große Netzwerke mit vielen gleichartigen PC-Arbeitsplätzen beschränkt. Der Einsatz eines netzwerkbootenden PCs eignet sich auch ideal für Embedded Systems[20].
Embedded Systems werden wegen ihrer kompakten Ausmaße, des geringen Wartungsaufwandes und der vielfältigen Einsatzmöglichkeiten an Bedeutung zunehmen. Sind größere Gerätezahlen für einen bestimmten Einsatzzweck geplant, kann es sich lohnen, auf lokalen Festspeicher zu verzichten und die Geräte aus dem LAN zu booten. Das LAN ist für viele Einsatzgebiete zwingende Voraussetzung, so dass die Funktionsfähigkeit dieser Geräte mit dem Funktionieren des Netzwerkes verknüpft werden darf.
In vielen Bereichen des wissenschaftlichen Rechnens wird bei Aufgabenstellungen, die sich gut parallelisieren lassen auf Cluster zusammengeschalteter PC-Systeme gesetzt. Diese bieten inzwischen eine sehr hohe Rechenleistung zum günstigen Preis. Damit die Kosten für das Betriebssystem nicht ausufern (die Zahl der Knoten geht meistens über die einhundert Computer), wird häufig Linux eingesetzt. Da die Installation einer entsprechend großen Zahl von Maschinen einheitlich und problemlos sein soll, bietet sich der Einsatz von Diskless Clients an, deren Filesystem an die speziellen Aufgaben angepaßt wird. So verringert sich neben dem Hardwareeinsatz (es kommt ja in erster Linie auf den Prozessor und notwendigen Arbeitsspeicher und nicht so sehr auf Festplatten und Installationslaufwerke an) der Administrationsaufwand einer solchen Anlage.
Auf diese Einsatzfelder soll jedoch im Rahmen dieser Arbeit nicht eingegangen werden.

6.5.5 Technologie der Thin-Clients

Die Grundlage beider Systeme ist identisch: Der PC wird mit einem Boot-ROM für BOOTP/DHCP und TFTP/NFS ausgestattet und startet den angepaßten Linuxkernel über das Netz. Die vorgestellten Thin-Clients verwenden eine Ramdisk und das Rootfilesystem wird von einem Server über NFS[21] bereitgestellt. Alle Netzwerk- und Konfigurationsdaten werden durch

[20] Dieses sind zumeist kleine PC-Systeme, die nicht als Arbeitsplatz-PC dienen, sondern für speziellen Aufgaben, wie Messwerterfassung oder Gerätesteuerung genutzt werden

[21] Network File System, ursprünglich von SUN Microsystems entwickelt und trotz vieler Probleme (Sicherheit und Performance) das Standardnetzwerkfilesystem, siehe hierzu auch Kapitel 8.4

BOOTP[22] oder DHCP[23] an den Thin-Client übertragen. Alle notwendigen Netzwerkeinstellungen und die Konfigurationen für Hard- und Software werden aus diesen Daten durch Skripten erzeugt. Diese Skripten orientieren sich an der Runlevel-Struktur klassischer Workstations, wobei jedoch mehr (X-Terminals) oder weniger (Diskless X-Stations) starke Reduktionen und Änderungen für den Bereich der Netzwerk- und Hardwarekonfiguration vorgenommen werden. Die Netzwerkdaten liegen bereits im Augenblick des Bootens des Kernels vor, da sie für ein Rootfilesystem basierend auf NFS unentbehrlich sind. Viele Hardwareparameter werden per DHCP übermittelt und sind nicht statisch im Filesystem hinterlegt (z.B. die XFree86-Konfiguration), sondern werden dynamisch beim Start erzeugt.
Linux Thin-Clients bieten über den reinen grafischen X11-Betrieb hinaus die Möglichkeit, mittels Citrix' ICA-Client[24] Verbindung zum Windows-Terminalserver aufzunehmen. So können Linux-Thin-Clients die Grundlage für Windowsclients bieten oder die Verknüpfung beider Welten in heterogenen Umgebungen darstellen.

[22] Bootprotocol

[23] Dynamic Host Control Protocol, welches eine Erweiterung des BOOTP darstellt und dieses fast überall abgelöst hat

[24] http://www.citrix.com, Windows Client-Server-Implementierung

Kapitel 7

Installationsüberlegungen

7.1 Überblick

In den folgenden Kapiteln soll das Installationsverfahren für die gewählte Client-Server-Architektur in ihren wesentlichen Schritten verdeutlicht werden. Das Konzept umfaßt drei aufeinander aufbauende Problemkreise, die sich aus der zentralen Anforderung ergeben, ein System von (zunächst) vollständig netzwerkfähigen Rechnerplätzen über einige zentrale Server implementieren zu können und gegebenenfalls Feinabstimmungen in unterschiedlichen Konfigurationen vorzunehmen.
Folgende Installationsschritte werden in den nächsten drei Kapiteln nacheinander in einer verwendbaren Basis-Lösung vorgestellt:

- Zu Beginn soll die Auswahl und die Implementierungsmöglichkeiten der benötigten Netzwerk- und Übertragungsprotokolle nachvollziehbar dargestellt werden. Diese Protokolle regeln den folgenden Bootvorgang und im weiteren den Datenverkehr zwischen Servern und Clients und sind unterschiedlich leistungsfähig. Die wesentlichen Installationsschritte erfolgen auf den zentralen Unix-Servern, sind aber restringiert durch die einsetzbaren Potenziale auf Seiten der Clients

- Der zweite Schritt ist der Bootvorgang der Clients über das Netz. Hierzu bieten sich alternative Realisierungen (z.B. Disk-On-Chip Realisierungen, Dual-Boot-Umgebungen) an. Da im vorliegenden Fall auch die Administration und Kontrolle einzelner Hardware-Komponenten zentralisiert werden soll, bedarf es hier einer Lösung, bei der auf Client-Seite nur eine minimale Bootkonfiguration einzelner Komponenten - insbesondere der Netzwerkkarten - zur Verfügung stehen sollte.

- Nach dem Bootvorgang erfolgt das Hochfahren und die Anpassung der verwendeten Basissoftware. Hier muss nicht nur ein Standard-Startprogramm und gemeinsame File-Server zur Verfügung gestellt werden, sondern es geht auch darum, die Installation individueller Optionen für unterschiedliche Konfigurationen sicherzustellen. Hierzu gehören die hardwarespezifischen Anpassungen, wie die Generierung einer Beschreibungsdatei für den Grafikserver und die Installation der Treiber einer evtl. vorhandenen Audiokomponente, des System Management Busses (SMB) oder angeschlossener Zusatzkomponenten, wie Scanner, CD-Brenner etc.

7.2 Umfang der vorgestellten Lösung

7.2.1 Kernprojekt

Kernpunkt des hier vorgestellten Projektes ist die Einrichtung einer Client-Server-Architektur, die mit möglichst geringem Ressourcenaufwand installiert werden kann und zentral zu administrieren ist. Die im folgenden aufgeführten Installationsschritte beziehen sich auf die zentralen Punkte des entwickelten Konzepts, d.h. insbesondere auf die Konfiguration und Regulation der Clients. Einen wesentlichen Teil nimmt hierbei die Server-Client-Interaktion ein, die für den Startvorgang detailliert beschrieben wird. Dabei geht es zum einen um die Benennung der notwendigen Protokolle und ihre konkrete Umsetzung. Weiterhin folgt eine Abhandlung zur Anpassung typischer PC-Hardware für die Aufnahme der Bootsoftware. Je nach angestrebtem Einsatz, bzw. vorliegender Hardware, kommen für die Bootsoftware unterschiedliche Ansätze in Frage, die einzeln beleuchtet werden. Dabei wird dem in der vorliegenden Umgebung eingesetzten Tool ein umfangreicherer Teil eingeräumt. Dies gilt auch für die Einbindung des Tools in die Hardware.
Natürlich können dabei weder alle nötigen noch alle denkbaren Installationsschritte aufgeführt werden. So müssen selbstverständlich zunächst die Server als Linux-Rechner konfiguriert und installiert werden und die Hardware muss (Netze, Einzelteile) einwandfrei funktionieren. Mögliche Erweiterungen sind schließlich in beinahe beliebiger Zahl denkbar, so dass sich die hier vorgestellten Fälle eher als Beispiele bzw. Variablen verstehen lassen sollten, die eine Grundlage zu weiteren Entwicklungen, keineswegs aber einen abgeschlossenen Katalog angeben können. Trotzdem wird auf die beiden wichtigen Bereiche, die Anpassung der Software für die gegebene Hardware, d.h. in erster Linie die Erstellung der Konfigurationsdatei für XFree86 und das Laden aller benötigten (Kernel-)Treiber, sowie das Starten ausgewählter

Dienste umfassend eingegangen.

7.2.2 Testumgebungen

Da diese Arbeit aus der Anwendung in der Praxis heraus entstanden ist, sollte darauf hingewiesen werden, dass Einzelheiten wie Verzeichnis- oder Dateinamen, aber auch die verwendeten Hard- und Software-Komponenten in den folgenden Kapiteln dem vorliegenden Beispiel der Studierendenumgebung an der Universität Göttingen entnommen sind[1]. Die im Lauf der Projektzeit von 30 auf über 400 Maschinen ausgebaute Anzahl der Arbeitsplätze erlaubt einen tiefgehenden Einblick in die verschiedenen Aspekte der Administration und Überwachung des Betriebes. Hierdurch ergaben sich eine Anzahl neuerer Anforderungen, welche erfüllt werden mußten: Das Konzept der X-Terminals konnte nicht mehr alle Bedürfnisse der Benutzer erfüllen, der Zugriff auf Wechselmedien, der Anschluß von Peripheriegeräten und das gewünschte Abspielen von Audiodateien erforderte eine modifizierte Herangehensweise, die den Arbeitsplatztyp "Diskless X-Station" hervorbrachte. Dies überschnitt sich mit der Weiterentwicklung der Leistung des zur Verfügung stehenden Hardwareangebotes, so dass gleichzeitig eine angemessene Ausnutzung der Ressourcen erzielt werden kann. Aufgrund der jeweils min. 150 eingesetzten Maschinen eines jeden Betriebstypes sind konsolidierte Aussagen möglich.
Es gibt eine Reihe weiterer Einrichtungen, die auf die Erfahrungen, welche im Zuge dieses Projektes gewonnen wurden, zurückgegriffen haben. Dazu zählen die Mathematische Fakultät mit einer Reihe von Studierenden- und Dozententerminals. Hier war zu Projektbeginn in erster Linie eine günstige Ersatzlösung für die auslaufende Linie älterer kommerzieller X-Terminals mit Schwarz-Weiß-Bildschirm gefragt. Die Neuausrichtung des wissenschaftlichen Rechnens mit einer stärkeren Einbindung des Betriebssystems Linux und der Verwendung der PC-Architektur auch für rechenintensive bzw. Server-Aufgaben führt jedoch inzwischen zu einer weitergehenden Berücksichtigung der Ergebnisse dieses Projektes.
Die Niedersächsische Staats- und Universitätsbibliothek setzt ihrerseits den Typ X-Terminal in größerem Umfang zur Ersetzung der älteren textbasierten Recherchesysteme ein. Ziel war hier, eine kostengünstige Lösung zu implementieren, die dem erweiterten Medienangebot Rechnung tragen soll. Das Anforderungsprofil liegt hier ebenfalls etwas anders als beim zugrundegelegten Projekt, so dass auch hier in erster Linie bereits im Kernprojekt gewonnenes Wissen angewandt wird.

[1] Siehe auch Anmerkungen in Kapitel 4

7.2.3 Besonderheiten

Software Der Betrieb einer großen Zahl rechnerbasierter Arbeitsplätze erfordert zwingend die Berücksichtigung einer ganzen Reihe von Vorgaben und Umgebungsbedingungen, welche teilweise recht speziell ausfallen können. Jene besonderen Umstände in allen Einzelheiten zu erläutern, würde den Fokus der Arbeit sprengen und von den entscheidenden Aspekten ablenken. In diese Kategorie gehören alle Anpassungen von Benutzerapplikationen und Systemwerkzeugen an den festplattenlosen Betrieb. Die meisten dieser Veränderungen gegenüber der Standardkonfiguration, wie sie vom Distributor vorgesehen wird, geschieht üblicherweise im Rahmen der notwendigen Site-Anpassungen, so dass diese selten besonders hervorstechen und behandelt werden müssen. Eine ganze Reihe von Erfahrungen sind in das vorgestellte Projekt eingeflossen, welche an verschiedenen Stellen wirksam werden. Die einfache "Weitervererbung" dieser spezifischen Einstellungen auf bestehende bzw. neu zu installierende Systeme, war eines der Anliegen, welche zur Beschäftigung mit automatischer Installation und vereinfachtem Systemupdate bzw. -abgleich führte.
In einigen Fällen ist die Lage von Konfigurations- bzw. Log-Dateien zu verändern, um den Bedingungen des festplattenlosen Betriebes auf Basis des Network File System sinnvoll zu genügen. Für den Betrieb von Diskless X-Stations mit weitgehendem Zugriff auf das Serverdateisystem erfolgen einige Anpassungen im Rahmen des Betriebes der grafischen Oberfläche. So sind vom Server her normalerweise die Anpassungen auf eine spezielle Hardware statisch. Der Einsatz von Thin-Clients erfordert jedoch eine breitere Palette zu unterstützender Hardware und dynamischere Konfigurationsmöglichkeiten. Die inzwischen weitestgehende Umsetzung des Filesystem Hierarchiestandards[2] bei modernen Linuxdistributionen erleichtert das Vorgehen jedoch, da eine der Designgrundlagen zur Verteilung der verschiedenen Dateien die Berücksichtigung des Netzwerkbetriebes ist.

Hardware Im Zusammenhang mit den Einstellungen der Software ergeben sich einige Anforderungen an die Hardwarekonfiguration, welche den späteren Betrieb stark erleichtern können. So sollte identische Hardware, z.B. ISA- bzw. PnP-Soundkarten, einheitlich konfiguriert sein. Weiterhin hat es sich als sinnvoll herausgestellt, Wechsellaufwerke nach einem festgelegten Schema an z.B. die IDE-Schnittstelle anzuschließen: Im besprochenen Kontext sind das CD-ROM-, bzw. LS120-Laufwerk einheitlich als "Master"

[2] FSHS: Die Entwicklungsgemeinde der freien (im Sinne diverser Lizenzen analog zur GPL) Unixe haben eine Empfehlung zur Anordnung von Dateien im Dateisystembaum verabschiedet. Einen Überblick zur Aufteilung der verschiedenen Dateiarten über den Linux-Dateibaum geben z.B. die Monografien [M5], [M6] und [M7].

am zweiten IDE-Controller angeschlossen, ZIP-Laufwerke als "Slave". Dadurch können aufwändige Abfrageroutinen und Skriptverzweigungen vermieden und die Hardwareadministration stark vereinfacht werden.

7.2.4 Einschränkungen des Fokus

Aus dem Massenbetrieb von Rechnern mit hohen Benutzerzahlen ergeben sich Anforderungen, die vom Desktop-Einsatz an Einzelplätzen abweichen. Dieses spezielle Problem weist jedoch über den Ansatz dieser Arbeit hinaus und beschäftigt sich mit der Tauglichkeit des Betriebssystems Linux für solche Aufgaben. Zur Rolle von Linux in der Riege ernstzunehmender Konzepte kann hier keine eingehende Diskussion erfolgen, sondern nur Puzzleteile beigetragen werden, die dessen Einsatzfähigkeit unterstreichen und Besonderheiten herausstellen, welche letztendlich zur Auswahl geführt hatten.
Mit der Netzwerksicherheit wäre ein weiteres Thema zu behandeln, welches jedoch gleichfalls über den Fokus des gewählten Ansatzes hinausgehen würde. Anmerkungen zu dieser Fragestellung geschehen am Rande, so sie besondere Aufmerksamkeit genießen sollten. Jedoch wäre auch diese Fragestellung in einer separaten Abhandlung zu würdigen.
Nicht speziell berücksichtigt werden weiterhin die Standardserver für FTP, WWW, Backup, Mail bzw. Benutzerdaten und -authentifizierung, etc., wie sie üblicherweise zum Betrieb eines Rechenzentrums bzw. des vorgestellten Projektes selbstverständlich sind und die zu weiten Teilen bereits unter Linux implementiert sind. Die Fragestellung der Userauthentifizierung und die Verwendung geeigneter Protokolle, die sich zum Teil mit dem Thema Netzwerksicherheit überschneidet, wird hier ausgeklammert. Aufnahme hingegen finden die oben genannten Rechner zumeist in der Datenbank zu Zwecken der Inventarisierung bzw. buchhalterischen Abrechnung, sowie ihrer Erfassung für den Domain Name Service und die Systemüberwachungstools.
Der Horizont dieser Arbeit geht von einer recht generischen Linuxdistribution aus, welche sich an die üblichen Vorgaben, wie Aktualität, Umfang und Konformität zum Filesystem Hierarchie Standard und der Linux Standard Base, hält. Zum Einsatz kommt die SuSE-Distribution in ihrer jeweils aktuellen Fassung, so dass einige Anpassungen spezifischer Natur sind, jedoch leicht für andere Distributionen abstrahiert werden können. Die Vor- und Nachteile der Auswahl spezieller Linuxzusammenstellungen kann der Fachliteratur, z.B. dem "Linux-Magazin", "iX" oder der "CT" entnommen werden.

Kapitel 8

Einrichten des Servers

8.1 Auswahl der notwendigen Netzwerkprotokolle

Nach der grundsätzlichen Betrachtung der einsetzbaren Client-Typen soll es in den folgenden Abschnitten um die Einrichtung des Servers für diese Geräte gehen. Hierzu sind Erläuterungen zu einigen Internetprotokollen und ihrer Implementierung unter Linux erforderlich. Einen guten Überblick über die TCP/IP-Suite verschafft z.B. die Abhandlung von Comer[1].
Zur Umsetzung der angestrebten Client-Server-Architektur werden einige Netzwerkprotokolle verwendet. Diese Dienste werden im folgenden etwas näher beschrieben und es wird kurz auf ihre Konfiguration, bzw. Implementierung eingegangen.
Der Server stellt BOOTP bzw. DHCP, TFTP, das Root- und Zusatzfilesysteme per NFS[2] und die allgemein benötigte Software bereit. Unter bestimmten Umständen kann auf TFTP verzichtet und dieses durch NFS zum Übertragen des Bootkernels ersetzt werden. Das Rootfilesystem der Clients liegt in einem eigenen Dateibaum, in der beschriebenen Beispielimplementation unterhalb */nfsroot/dxs* für Diskless X-Stations und in */nfsroot/dxt* für die X-Terminals. Auf dem Server werden über Eintragungen in der Konfigurationsdatei des **dhcpd** die Thin-Clients weitestgehend konfiguriert. Hierfür können eigene Optionen, die über die bereits vordefinierten hinausgehen, in gewissem Umfang frei gewählt werden.
In den nächsten beiden Kapiteln sollen die Gemeinsamkeiten in der Server- und Clientkonfiguration beider Terminaltypen betrachtet werden. Auf die

[1] siehe [M1] oder auch [M2] von Dawson und Kirch
[2] Network File System

Unterschiede wird in einem weiteren Kapitel eingegangen werden. Dabei wird die Beschreibung der Dienste in der Reihenfolge des Bootvorganges erfolgen.

8.2 Das Boot-Protokoll

BOOTP und sein Nachfolger DHCP ist ein UDP-basiertes Netzwerkprotokoll[3], über das grundlegende Daten zur Konfiguration eines Clientsystems übertragen werden können. Es arbeitet auf Port 67 zur Anfrage an den Server und auf Port 68 zur Rückantwort an den Client. Dieses Protokoll läßt sich nur sinnvoll in broadcastfähigen Netzen[4] einsetzen. Es ist nicht routebar, aber erreichbar um einen Gatewaydienst einzurichten. Mittels BOOTPGW[5] gelingt es, BOOTP-Pakete in einem Subnetz zu registrieren und an einen spezifizierten Server weiterzuleiten. Der Nachteil liegt jedoch darin, dass die Redundanz, welche durch mehrere Bootserver erreicht werden kann, entfällt. Die einzelnen Parameter zur Konfiguration werden nachfolgend im Zusammenhang mit den beiden existierenden Implementierungen erläutert. Ein Teil der Informationen, die bereitgestellt werden, übernimmt der Kernel schon durch den BOOTP-Prozess, der von der Bootsoftware gestartet wird. Hierzu gehört die IP-Konfiguration mit Netzmaske, Hostname und Standardgateway. Die Parameter zum DNS werden später gesondert angefragt.

8.2.1 Die Implementierung als "bootpd"

Der BOOTP-Dämon ist die ältere verfügbare Software und wird hier wegen ihrer Einschränkungen nur der Vollständigkeit halber angeführt. So muss z.B. bedacht werden, dass mit dem BOOTP-Paket[6] nur eine bestimmte Menge an Informationen transferiert werden kann. So können nicht alle dargestellten Optionen auf einmal übertragen werden. Dieses läßt sich jedoch über sogenannte BOOTP-Extensions realisieren. In diesem Fall bietet sich auch deshalb die Verwendung des DHCP an, welche im nächsten Abschnitt beschrieben wird. Im Anhang[7] findet sich daher nur eine kurze Erläuterung seiner Konfiguration.

[3] RFC951 : "Bootstrap Protocol (BOOTP)", September 1985

[4] üblicherweise im LAN Ethernet oder TokenRing. Serielle Point-to-Pointverbindungen nutzen zur dynamischen IP-Konfiguration üblicherweise PPP (Point-to-Point-Protocol)

[5] Es existieren sowohl für BOOTP als auch DHCP entsprechende Tools, um diese Gatewayfunktionalität zu implementieren. Auf jedem besseren TCP/IP-Router kann dieser Dienst aktiviert werden.

[6] üblicherweise 572 Byte incl. Header für IP und UDP

[7] Siehe hierzu Abschnitt C.2

8.2.2 Die Implementierung als "dhcpd"

DHCP[8], das "Dynamic Host Control Protocol" stellt eine Erweiterung und Ergänzung des BOOTP dar. Weiterentwicklungen, wie z.B. dynamisches DNS, finden in erster Linie beim **dhcpd** statt, die verwendeten **bootpd** Implementierungen stammen aus den Anfängen der 90er Jahre und werden nur noch im Fall grober Sicherheitsverletzungen gepatcht. Das Internet-Software-Consortium[9] entwickelt eine Beispielimplementation des DHCP und einiger anderer Internetprotokolle. Der Sourcecode liegt in den meisten Fällen auch in einer Linuxanpassung vor, so dass eine einfache Übersetzung für die Zielplattform erfolgen kann.
Die Einschränkungen im BOOTP einerseits und seine anerkannten Vorteile zur Konfiguration großer Rechnerzahlen andererseits haben zu einer Weiterentwicklung geführt. Sollte es notwendig sein viele Optionen an die Clientsysteme zu übertragen, empfielt sich der Einsatz des **dhcpd**, der auch über eine umfangreichere Palette an vordefinierten Optionen verfügt. Für Server, die eine Mischumgebung aus Windowsystemen mit dynamischer IP-Zuweisung und X-Terminals bestehen, kommt nur noch DHCP in Frage, da das "alte" BOOTP diese Funktionalität nicht bereitstellt.
DHCP wird aufgrund seiner Flexibilität zum zentralen Konfigurationstool. Es wird versucht werden, über diesen Dienst alle relevanten Informationen zum Betrieb von Linux Thin-Clients zu übertragen. Neben den klassischen Parametern, wie Hostname, IP-Adresse, Netzmaske und Gateway, zählt dazu eine Reihe von Server-IP's: X-Display-, Time-, Swap-, NIS-Server etc. Darüberhinaus soll in einem Stringfeld die Konfiguration von Grafikkarte und Monitor einschließlich Mausanschluß übermittelt werden. Ein weiteres Feld nimmt Kommandozeilen auf, die an die Datei */etc/init.d/boot.local* angehängt und ausgeführt werden. Durch dieses Vorgehen lassen sich die Audiokomponenten konfigurieren oder zusätzliche Gerätetreiber laden.
Im Weiteren bezieht sich daher die Beschreibung auf die DHCP-Implementierung des ISC. Diese liegt inzwischen in der dritten Version vor und unterstützt eine ganze Reihe neuer Eigenschaften[10]. Dazu zählen z.B. Use-

[8] Die grundlegenden RFCs für DHCP sind:

- RFC1541 : "Dynamic Host Configuration Protocol", Oktober 1993
- RFC2131 : "Dynamic Host Configuration Protocol", März 1997
- RFC2132 : "DHCP Options and BOOTP Vendor Extensions", März 1997

[9] siehe dazu [W6], wobei das ISC neben der DHCP-Implementierung weitere Internet-Standards betreut.

[10] Die Erweiterungen zum DCHP:

- RFC2485 : "DHCP Option for The Open Group's User Authentication Protocol", Januar 1999

roptionen, Vendor-Code-Identifier[11], dynamisches DNS[12], zeitbeschränkte Zuweisungen der Konfigurationen (sogenannte Leases). Eine Beispielkonfiguration und die Erläuterungen zu wichtigen Parametern und deren Eigenschaften sind dem Anhang (Abschnitt C) sowie den Ausführungen von Droms/Lemon[13] zu entnehmen.

8.3 Das TFT-Protokoll

Das Trivial File Transfer Protocol (TFTP)[14] dient zur Übertragung des zu bootenden Netzkernels[15] bzw. eines PXE-Bootimages[16] zu einem sehr frühen Zeitpunkt. Das Protokoll wird zu Beginn des Bootvorganges ausgeführt und muss neben der BOOTP-Implentation auf einem 16 kByte bzw. 32 kByte EPROM Platz finden. Deshalb ist es auf geringen Codeumfang optimiert. Weitere Ausführungen zum Aufbau der Bootsoftware finden sich im nächsten Abschnitt.
TFTP bietet keinen Schutz durch Passwortauthentifizierung und sollte deshalb mit Vorsicht, d.h. mindestens durch Beschränkung des Hauptverzeichnisses auf den benötigten Bereich aufgesetzt werden. Dazu kann zum einen durch den **(x)inetd** ein TFTP-Root angegeben werden, was einen Zugriff auf höhere Verzeichnis-Ebenen des Servers verhindert. Zur Illustration dienen die nachfolgenden Ausschnitte aus den jeweiligen Konfigurationsdateien. Der folgende Ausschnitt zeigt ein Teil der */etc/inetd.conf*.

- RFC2489 : "Procedure for Defining New DHCP Options", Januar 1999

[11] Diese ermöglichen eine Identifizierung von Clients und das verwenden spezieller Strings in der DHCP-Antwort

[12] in Zusammenarbeit mit dem ISC DNS-Daemon

[13] siehe hierzu [M3]

[14] TFTP wird in den folgenden RFCs beschrieben:

- RFC1350 : "The TFTP Protocol (REVISION 2)", Juli 1992
- RFC2090 : "TFTP Multicast Option", Februar 1997
- RFC2347 : "TFTP Option Extension", Mai 1998
- RFC2348 : "TFTP Blocksize Option", Mai 1998
- RFC2349 : "TFTP Timeout Interval and Transfer Size Options", Mai 1998

[15] Welche Optionen dieser enthalten muss, welche Anpassungen notwendig sind und wie man ihn mit einer entsprechenden Bootsignatur versieht, wird in den folgenden Abschnitten erklärt

[16] Die notwendigen Erweiterungen von TFTP im Zusammenhang mit PXE finden sich im RFC2349

```
    [...]
    #
    # Tftp service is provided primarily for booting.
# Most sites run this only on machines acting as
# "boot servers".
    #
    tftp dgram udp wait nobody /usr/sbin/tcpd \
in.tftpd /boot
    bootps dgram udp wait root /usr/sbin/bootpd \
bootpd -c /nfsroot
    #
    [...]
```

Oder: (Ausschnitt der */etc/xinetd.conf*)

```
    [...]
    # disabled = bootps
    # disabled = tftp
    [...]
    service tftp
    {
    socket_type = dgram
    protocol = udp
    wait = yes
    user = nobody
    only_from = 134.76.0.0/16 10.0.0.0/8
    server = /usr/sbin/in.tftpd
    server_args = /nfsroot
    }
```

8.4 NFS - Das Networkfilesystem

Der BOOTP bzw. DHCP-Server wird in den meisten Fällen gleichfalls das Rootfilesystem für die Clients zur Verfügung stellen. NFS[17] stellt die am häufigsten verwendete Variante zur Verteilung von Filesystemen über TCP/IP-Netze dar. NFS liegt inzwischen in den Versionen 2 und 3 vor. Die Unterstützung für dieses Filesystem muss sowohl im Serverkernel als auch im Kernel der Clients aktiviert sein. Im Kernel des Servers sollte für Test- und Überwachungszwecke auch die Clientunterstützung aktiviert sein, der Client-Kernel wird auf Serverdienste verzichten können oder kann diese in ein Modul auslagern.

[17] u.a. beschrieben im RFC1094

Im Kernel für die Clientsysteme wird dafür die Option "[*] Root file system on NFS" aktiviert (weitere Optionen finden sich in Kapitel 10.2). Damit versucht das Terminal nach dem Start des Kernels sein Filesystem vom Server einzubinden[18]. Damit dieses gestattet wird, sollten die beiden NFS-Daemonen (**rpc.nfsd** und **rpc.mountd**) auf dem Server installiert und aktiviert sein. Unter Linux stehen inzwischen zwei Implementierungen zur Verfügung, die klassischen Userspace-Daemonen und die direkte Kernelunterstützung. Unabhängig vom eingesetzten Typ werden in der */etc/exports* des Servers die Freigaben eingetragen:

```
    # Rootfilesystem der X-Terminals
    /nfsroot/dxt 172.16.0.0/255.255.0.0(ro,no_root_squash)
    # Rootfilesystem der Diskless X-Stations
    /nfsroot/dxs 172.16.0.0/255.255.0.0(ro,no_root_squash)
    # /usr - Freigabe für DXS
    /usr 172.16.0.0/255.255.0.0(ro)
    # /opt - Freigabe für DXS
    /opt 172.16.0.0/255.255.0.0(ro)
    # /tmp/users - Freigabe für DXS (schreibbar für \
# groessere Userdaten zur Entlastung der Ram-Disk)
    /tmp/users  172.16.0.0/255.255.0.0(rw)
```

Im Beispielfall befindet sich der NFS-Root-Baum unter */nfsroot/dxt* für die X-Terminals und unter */nfsroot/dxt* für die Diskless X-Stations. Für letztere werden zusätzlich weitere Bereiche des Serverfilsystems hinzugemountet, was im Kapitel 11 besprochen wird. Grundvoraussetzung hier ist jedoch die Verwendung einer einheitlichen Hardwarearchitektur, da die ausführbaren Programme und Bibliotheken gemeinsam verwendet werden. Aus Sicherheitsgründen wird das Filesystem nur ReadOnly exportiert, für Testzwecke empfiehlt sich jedoch die Einstellung ReadWrite mit Root-Zugriff `(rw, no_root_squash)`. Da eine ganze Reihe von Basisprogrammen eingeschränkte Benutzerrechte hat, sollte zumindest das Rootfilesystem mit der Option "no_root_squash"[19] freigegeben werden. Die vom Server direkt übernommenen Teile des Dateibaumes sind gegebenenfalls in ihren Benutzerrechten anzupassen oder die entsprechenden Programme an eine geeignete Stelle zu verschieben bzw. zu verlinken. Trotz der weitestgehenden sauberen und durchdachten Aufteilung der verschiedenen Dateiarten (Programme, Bibliotheken, Konfigurationsdateien, Dokumentation, ...) auf den

[18] siehe hierzu auch die **mknbi-(linux)** -Optionen bzw. DHCP-Einstellungen (Abschnitt B.4)

[19] Aus Sicherheitsgründen werden die Dateien mit Benutzerrechten des Systemadministrators auf die Benutzerkennung des "nobody" (Standardnutzer mit geringsten Rechten) umgemappt. Dieses läßt sich durch die angegebene Option unterbinden.

Filesystem-Baum der jeweils gewählten Linux-Distribution wird es an einigen Stellen notwendig sein, Modifikationen gegenüber einer Standardinstallation vorzunehmen. Dies wird besonders im Verzeichnis für die variablen Daten (*/var* mit Log-, Lock-, Spool-, Cache-Dateien etc.) der Fall sein. Teile hiervon werden im Schreibzugriff benötigt, jedoch sollte vermieden werden, die Ramdisk durch Überfrachtung zu stark anwachsen zu lassen.
Anmerkung: Wenn als NFS-Server eine abweichende Hardware- bzw. Software-Plattform zum Einsatz kommt, ist zu überprüfen, ob das Deviceverzeichnis */dev* der Terminals komplett korrekt mit Major- und Minornummern[20] übertragen wird. In einer Umgebung aus DEC-Alpha-Stations (DEC-Unix und RedHat-Linux) gelang dieser Export bisher nicht sauber. Hier kann das Devicefilesystem weiterhelfen (Unterabschnitt 10.3), welches seit einiger Zeit mit den neueren Kernelversionen verteilt wird.

[20]Die Spezialdateien im Device-Verzeichnis stellen eine Schnittstelle zum Kernel dar, addressiert werden sie über Haupt- und Untergerätenummern

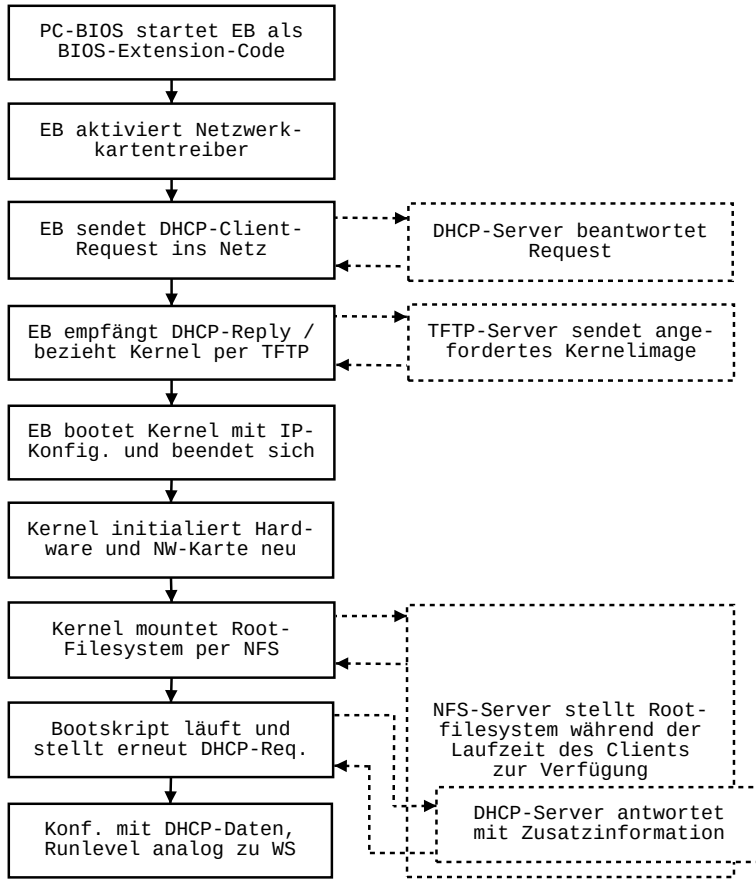


Abbildung 8.1: Client-Server-Interaktion beim Booten

Kapitel 9

Einrichtung der Client-Hardware

9.1 Vorüberlegungen

Die nun folgenden Abschnitte beschäftigen sich mit der Frage, wie die Thin-Clients in die Lage versetzt werden können, über ein Netzwerk zu booten. Thin-Clients sollen den Administrationsaufwand senken, dabei aber gleichzeitig eine gewisse Auswahl an Hardware unterstützen, da sich der Einsatz der jeweils nächsten PC-Generation schrittweise vollzieht. In den wenigsten Fällen wird man den Gesamtbestand der Arbeitsplätze einer Organisation in einem Zuge austauschen. Die zu verwendende Bootsoftware soll deshalb nicht aufwändig zu konfigurierende und teurere Spezialhardware voraussetzen, sondern möglichst die Gegebenheiten der verwendeten PC-Hardware ausnutzen.
Die Bootsoftware muss nach dem Ausschalten des Gerätes wieder verfügbar und automatisch ohne besondere Benutzerinteraktion im Standardbetrieb aktivierbar sein. Dabei können spezielle Anforderungen, wie z.B. Dual-Boot-Lösungen, etwas andere Implementierungen erfordern. Alle Ansätze sollten sich jedoch aus Sicht der Software auf dem Server möglichst identisch verhalten, um den Aufwand spezifischer Anpassungen gering zu halten[1].
In diesem Kapitel geht es daher um die Auswahl der Bootsoftware, welche über die Aufgabenverteilung zwischen Client und Server bestimmt und den Hardware- und Administrationsaufwand für die einzelne Maschine festlegt. Anschließend geht es um konkrete Erfordernisse an einzelne Hardwarekom-

[1] Dieses beträfe z.B. unterschiedlich aufzubereitende Kernel für die Zusammenarbeit mit Etherboot bzw. PXE und dem Syslinux-Paket.

ponenten, um die Bootsoftware dauerhaft auf dem Client zu installieren. Hier kommen verschiedene Varianten in Betracht, welche in ihren Vor- und Nachteilen beleuchtet werden. Dem klassischen Weg - die Installation eines Boot-ROM's auf der Netzwerkkarte und die Auswahl der geeigneten Chips - wird die Modifikation des Mainboard-BIOS als Alternative gegenübergestellt. Neuere Kompaktrechner und Netzwerkkarten stellen inzwischen PXE zur Verfügung. Dessen Verwendbarkeit wird im Unterkapitel 9.2.7 besprochen. Weitere Bootmedien, wie z.B. vorhandene Festplatten, können je nach Aufgabenstellung genutzt werden. Es folgen deshalb einige Bemerkungen zu Dual-Bootlösungen, d.h. die Auswahlmöglichkeit zwischen verschiedenen Betriebsmodi eines Rechners[2]. Mit der Installation der Bootsoftware auf den Clients ist der hardwareseitige Teil der Konfiguration abgeschlossen.

9.2 Bootsoftware

9.2.1 Auswahl der Software

Es gibt eine ganze Reihe von Möglichkeiten, Thin-Clients zu betreiben: Einige Hersteller setzen eine Solid State Disk oder Flash-Speicher für ein Minimalbetriebssystem ein. Dieses stellt zwar eine gewisse Unabhängigkeit von der Verfügbarkeit des Servers oder Netzwerkes her, erfordert aber einen höheren Hardwarebedarf und Kostenaufwand. Bei einer steigenden Zahl von Maschinen steigt der Aufwand bei Updates und Rekonfigurationen. Weiterhin sollte das Wachstum des Software-Umfanges nicht unterschätzt werden, welches recht schnell dazu führt, dass vorhandene Festspeicherkapazitäten nicht mehr ausreichen.
Um die beschriebenen Probleme zu vermeiden, wird eine Lösung angesetzt, die fast alle Funktionalität auf die Serverseite verlagert und nur eine Minimallösung für den Client erfordert. Damit sinkt die Fehleranfälligkeit auf der Seite der Thin-Clients mit steigenden Anzahlen von Maschinen.
Die Aufgabe der Bootsoftware wird darauf beschränkt, den Client nach dem Start mit einer minimalen Netzwerkfähigkeit auszustatten, um alle weiteren Daten von einem oder mehreren Servern beschaffen zu können.
Der anfängliche Startvorgang erfolgt analog zu Diskless X-Stations und X-Terminals. Das Boot-Protokoll und als Weiterentwicklung das DHCP gestatten die automatische Netzwerk- und Bootkonfiguration von Clientsystemen. Daher müssen keinerlei Informationen zu IP-Nummer, Netzmaske, Hostna-

[2] Dieses kann bedeuten, dass unterschiedliche Software, z.B. ein Windowsbetriebssystem von der Festplatte und eine Linuxumgebung aus dem Netz auf einem Rechner zur Auswahl gestellt wird. Weiterhin könnten auch unterschiedliche Konfigurationen desselben OS auf diese Weise gestartet werden.

me etc. auf dem Clientrechner bekannt sein. Das ermöglicht eine einfache Anpassung bei Änderungen in der Netzwerkstruktur und erlaubt festspeicherlose Clients und einen leichten Wechsel eines Gerätes zwischen verschiedenen Netzen und Betriebsmodi. Die eleganteste Lösung für Thin-Clients besteht in der Installation eines Boot-EPROMs auf der Netzwerkkarte oder in einem BIOS-Patch. Eine kleine Bootsoftware, die auf der Clientmaschine installiert ist, sendet BOOTP- bzw. DHCP-Broadcastpakete in das lokale Netzwerk und erhält zuerst ihre IP-Konfiguration und dann einen Bootkernel per TFTP übertragen. Hierfür werden die Terminals anhand ihrer Hardwareadresse der Netzwerkkarte (MAC-Adresse) identifiziert.

9.2.2 Anforderungen an die Netzwerkkarten

Fast alle gängigen Netzwerkkarten werden vom Etherboot- oder Netboot-Paket unterstützt. Dazu zählen die NE2000 und kompatiblen in der ISA- und PCI-Version, bis hin zu den 100Mbit-Typen mit dem RTL8139, 3COM, Via-Rhine, SiS, SMC-Ultra-Karten, Davicom 9102, IntelEtherExpress oder Tulip (und kompatible) Chipsets, die alten 3COM 503 und 509, die WD8003 und 8013 kompatiblen (die genannten Typenbezeichnungen bilden das Spektrum populärer Netzwerkkarten und ihrer Chipsets ab). Weitere Modelle sind einsatzfähig, wenn sie über einen mitgelieferten Packettreiber des jeweiligen Hardwareherstellers verfügen.
Weiterhin sollte die Netzwerkkarte einen Sockel für das Boot-ROM enthalten und eine ROM-Größe von 16 kByte bzw. 32 kByte unterstützen. Mit einigen Einschränkungen läßt sich der Code auch für ein 8 kByte ROM erstellen. Sehr gute Erfahrungen ergeben sich mit fast allen eingesetzten Netzwerkkarten, unabhängig von ihrer ISA- oder PCI-Ausführung. Jedoch können immer wieder Treiberprobleme, wie z.B. mit verschiedenen Tulip-Chip-Clones auftreten. Etherboot leitet seine Treiber von denen des Linux- bzw. BSD-Kernels ab, so dass in den meisten Fällen mit einer zügigen Umsetzung nach Erscheinen einer neuen Netzwerkkarte zu rechnen ist.
Ein alternative Herangehensweise wählt das Netbootpaket[3] von Gero Kuhlmann. Diese Software unterstützt alle Netzwerkkarten, die über einen Packettreiber (mitgeliefert oder freie Implementierung auch im Crynwr-Paket[4]) oder einen NDIS-Treiber verfügen. Hierzu erhält das EPROM einen kleinen Bootblock, der zuerst den Netzwerkkartentreiber und anschließend BOOTP und TFTP aufruft. Jedoch ist die Robustheit dieses Pakets nicht so hoch wie Etherboot einzuschätzen. Mit Netboot begibt man sich in die direkte Abhängigkeit von der Treiberunterstützung seitens der Hersteller. Die

[3] http://www.han.de/ gero/netboot
[4] zu beziehen über jeden größeren FTP-Server oder direkt ftp://ftp.crynwr.org

Verfügbarkeit dieser ist vor dem Einsatz zu prüfen.
Inzwischen gibt es erste Netzwerkkarten, wie die "3C905C" von 3COM, die über eine eingebaute BIOS-Erweiterung das Booten aus dem Netz über verschiedene Protokolle - neben IPX/SPX und PXE auch BOOTP/TFTP - beherrschen. Hier ist jedoch ein abweichendes Kerneltagging[5] notwendig. Die nächsten Abschnitte behandeln die zur Verfügung stehenden Alternativen, wobei auf die interessanteste, PXE, ausführlicher eingegangen wird.

9.2.3 Das Etherboot-Paket

Das Etherbootpaket wird von einer größeren Zahl von Programmierern entwickelt und enthält inzwischen die Treiberimplementierung fast aller gängiger Netzwerkkarten. Es unterstützt vielfältige Optionen, die ein Zusammenspiel mit anderen Bootloadern, Dual-Boot-Lösungen, Bootmenüs etc. erlauben. Die aus dem Etherbootpaket kompilierten Bootimages sind von sehr geringem Umfang und belegen zwischen 8 kByte und 64 kByte Speicher. Darin enthalten sind der Treiber für die Netzwerkkarte und die notwendigen Netzwerkprotokolle DHCP und TFTP bzw. NFS[6]. Der geringe Codeumfang erlaubt eine einfache Installation in einem EPROM auf dem jeweils verwendeten Netzwerkadapter oder als Patch im Mainboard-BIOS. Weiterhin ist es denkbar, Etherboot als ausführbare DOS-Datei zu erzeugen oder direkt in den Bootsektor einer Diskette zu schreiben, was einfaches Testen und Debugging ermöglicht.
Etherboot wird unter der GPL verteilt, erlaubt also eine beliebige Anzahl von Installation ohne Lizenzkosten. Durch die Offenlegung des Source-Codes besteht die Aussicht eigene Erweiterungen vorzunehmen oder Anpassungen an das eigene Netzwerk vorzusehen. Die Installation und Kompilation wird im Anhang in Teil B.2 beschrieben.
Durch die Beschränkung auf wenige Aufgaben kann die Fehleranfälligeit im Code des erzeugten Bootimages als gering eingestuft werden, was aufwändige Updates der jeweils betroffenen Rechner vermeiden hilft. Eine einmal erfolgreich geteste Installation muss jedoch beim Austausch der Netzwerkkarte gegen einen anderen Typ oder bei bestimmten Veränderungen der Konfiguration des DHCP, wie der Verschiebung der Portadressen oder die Benutzung von Vendor-Code-Identifiern geändert werden.
In neueren Etherboot-Implementierungen kann zum Laden des Bootkernels anstelle von TFTP auch NFS verwendet werden, da dieses anschließend

[5] Das Taggen, d.h. versehen mit einer geeigneten Bootsignatur, wird im Kapitel B.4 erläutert. In diesem Fall werden jedoch andere Software-Werkzeuge benötigt, die von 3COM (http://www.3com.com) bezogen werden können.

[6] Der Codeumfang bei der Auswahl von NFS erhöht sich geringfügig gegenüber dem Einsatz von TFTP und ist bei sehr kleinen EPROM-Größen sehr genau abzuschätzen.

für das Rootfilesystem zum Einsatz kommt. So kann auf Serverseite ein Dienst reduziert werden, womit sich die Ausfallwahrscheinlichkeit verringern und die Systemsicherheit erhöhen läßt. Wie Etherboot in diesem Fall zu konfigurieren ist, kann dem Anhang B.2 entnommen werden.

9.2.4 Das Netboot-Paket

In einigen Fällen bietet sich das Netboot-Paket[7] als Alternative zu Etherboot an. Falls für eine bestimmte Netzwerkkarte keine Treiberunterstützung vorliegt und gleichzeitig ein Treiber vom Hersteller existiert, kann Netboot eingesetzt werden. Es verfährt etwas anders als Etherboot, indem zuerst eine Umgebung geschaffen wird, in der die mitgelieferten DOS- bzw. NDIS-Treiber des Herstellers geladen werden können. Anschließend wird analog zu Etherboot der DHCP/TFTP-Prozess gestartet. Beide Pakete verwenden ein identisches Kerneltagging, womit eine einheitliche Kernelbasis für verschiedene Clients genutzt werden kann.
Die Weiterentwicklung dieses Projektes hat sich verlangsamt, da inzwischen eine sehr gute Unterstützung vieler Netzwerkkarten durch Etherboot vorliegt, welches inzwischen einfacher zu implementieren ist. Die Einrichtung und Anwendung des Paketes wird daher nur kurz im Anhang im Abschnitt B.3 abgehandelt.

9.2.5 Das NILO-Projekt

NILO steht für Network Interface Loader und soll das Booten von Linux/BSD und den Windowssystemen erlauben. Dieser Loader ist die Weiterentwicklung beider vorher beschriebenen Pakete und soll durch die Verwendung der Linux-Kerneltreiber jede von Linux unterstützte Netzwerkkarte benutzen können. Dieses Paket befindet sich in der Entwicklung und ist daher nicht in jedem Fall einsatzfähig. Zum derzeitigen Zeitpunkt bietet es keine Alternative zu Etherboot.

9.2.6 GRUB

GRUB[8] ist ein universeller Bootloader, der zusammen mit Etherboot neben dem üblichen Multiboot auch das Booten aus dem Netzwerk unterstützt. Dieser Bootloader verfügt über etliche Einstellungen und kann eine ganze Reihe verschiedener Betriebssysteme und Dateisysteme booten. Leider ist GRUB recht aufwändig in der Konfiguration. Da dieser Bootloader inzwi-

[7] Siehe hierzu http://netboot.sourceforge.net; Autor ist Gero Kuhlmann
[8] GRand Unified Bootloader

schen einer ganzen Reihe von Linuxdistributionen beiliegt, wird sich der Einsatzkomfort in Zukunft sicherlich verbessern.

9.2.7 Preboot eXecution Environment (PXE)

Intels PXE[9] stellt eine weitere Möglichkeit dar über das Netz eine plattenlose Maschine zu booten. Mittels DHCP/TFTP wird ein PXE-Image geladen, welches dann weitere Bootfunktionalität zur Verfügung stellt. Das Syslinux-Paket[10] enthält neben anderen Funktionen auch die Unterstützung für PXE. Dieses ist dann in der Lage "ungetaggte" Kernel per TFTP zu booten und zu starten. In diesen Kerneln muss jedoch BOOTP/DHCP aktiviert sein, da sich keine Parameter vom PXE übergeben lassen.
Weiterhin ist es vorstellbar neben Kernelimages Bootsektoren anderer Betriebssysteme zur Auswahl zu stellen und damit ein netzgesteuertes Multiboot zu erlauben.
Die Preboot Extension läßt sich inzwischen auch gemeinsam mit Etherboot verwenden. Es wird in diesem Fall ein Kettenstart ausgeführt, bei dem zuerst PXE aktiviert wird, ehe dieses ein auf PXE zugeschnittenes Etherboot lädt, das wiederum den weiteren Bootvorgang durchführt. Diese Kombination erweist sich dann als hilfreich, wenn PXE bereits vorhanden und ein Eingriff in die bestehende Hard- und Software nicht erwünscht ist. Ein weiterer Vorteil liegt hier in der Zentralisierbarkeit der Bootkonfiguration, da bei Veränderungen und Updates kein Austauschen der Bootsoftware auf den Clientsystemen mehr erfolgen muss. Dem steht jedoch nun ein etwas umfangreicherer Bootvorgang mit geringeren Eingriffsmöglichkeiten entgegen. Die beschriebene Anwendung setzt jedoch einen aufwändiger konfigurierten DHCP-Server voraus, da er zuerst die PXE-Anfrage und anschließend Etherboot mit den korrekten Daten versorgen muss. Hierzu muss der DHCP-Server anhand der Vendorcode-Identifier unterscheiden, welcher Client die Bootanfrage stellt und mit den entsprechenden Informationen antworten. Der notwendige Unterschied liegt zuerst in der Datei, die geladen wird: PXE benötigt für das Booten in Kette mit Etherboot die Angabe zur Etherbootdatei, Etherboot anschließend jedoch das klassische Kernelimage. Im Anhang (Abschnitt C.3.4) ist eine genaue Anleitung zur Konfiguration des DHCP-Servers und zum Erstellen eines Etherboot-PXE-Images angefügt.

[9] Die PXE-Option ist im BIOS einiger Motherboards mit onboard Netzwerkkarte und z.B. in der 3COM 905C enthalten.
[10] Syslinux stellt eine Art erweitertes **loadlin** dar

9.3 Unterbringung der Bootsoftware

9.3.1 Der Bootcode im Mainboardbios

Einige All-In-One-Mainboards (Audio, Netzwerkadapter) vermitteln bereits die Idee, was erreicht werden kann: Es gibt bereits eine BIOS-Einstellung - "Boot from Network", leider jedoch meist nur für ein IPX/SPX Netzwerk, manchmal auch für PXE. Dieses läßt sich jedoch in den meisten Fällen verändern oder hinzufügen, wie im folgenden erläutert wird.
Die meisten handelsüblichen neuen Mainboards verfügen über ein AWARD- bzw. AMI-BIOS, das in einem Flash-Speicher untergebracht ist. Mit einer speziellen Software läßt sich der Bootcode dem BIOS hinzufügen. In vielen Fällen kann so auf separate EPROMs und damit auf einen EPROM-Programmiergerät verzichtet werden.
Man benötigt jeweils zwei Software-Tools, eines zum Auslesen und Beschreiben des Speicherbausteins und eines zum Modifizieren des BIOS-Codes. Die Tools zum Programmieren des Flashs liegen üblicherweise dem Board bei oder sind einfach über AWARD[11] (hier awdflash.exe) oder AMI[12] (dann amiflash.exe) zu beziehen.
Auch unter Linux gibt es inzwischen einige (wenn auch eingeschränkte) Optionen: Das "/dev/bios"-Projekt[13] beschäftigt sich damit, über ein Kernel-Device auf verschiedene Flash-Bausteine zuzugreifen. So kann durch einfaches Laden des Moduls und Anlegen des Devices[14] mit "dd" das BIOS ausgelesen und geschrieben werden. Trotzdem ist hier noch Vorsicht angebracht, da nicht alle Mainboard-Chipsets und Flash-ROM-Größen vollständig unterstützt werden. Weiterhin werden Kernelmodule für generische Schnittstellen zu "Memory technology devices" entwickelt, welche in Zukunft die Ansätze des erstgenannten Projektes aufnehmen und weiterenwickeln könnten.
Zur Analyse und Modifikation eines AWARD-BIOS kann **cbrom.exe** verwendet werden, für ein AMI-BIOS **amibcp.exe**. Die beiden genannten Tools können zusäztlichen Code hinzufügen oder überflüssige bzw. in der beschriebenen Situation nutzlose BIOS-Komponenten entfernen. Eine genauere Anleitung zur Benutzung findet sich im Anhang (Abschnitt A). Auf diesen Wegen kann man Etherboot in das Mainboardbios integrieren, ohne überhaupt den Rechner öffnen zu müssen.

[11] http://www.award.com
[12] http://www.ami.com
[13] http://www.freiburg.linux.de/ stepan/bios: Das Kernelmodul konnte mit einigen Mainboards bereits erfolgreich erprobt werden. Dadurch konnte vermieden werden zum BIOS-Update Diskettenlaufwerke in die jeweiligen X-Terminals einbauen zu müssen.
[14] Characterdevice */dev/bios*, Major 104, Minor 0

9.3.2 (E)EPROMS

Lässt sich die eben beschriebene Variante wegen zu alten Boards/BIOS[15] oder zu geringem freien Speicher im Flash nicht anwenden, kann man auf seperate PROMs[16] ausweichen. Als Eproms können alte BIOS-Bausteine von Mainboards verwendet werden, so sie ein Fenster zum Löschen des Inhalts besitzen. Neu kosten die Bausteine je nach Händler, Größe und Art ca. 4 bis 10 DM.
Die Typenbezeichnung der ROM-Bausteine sind wie folgt zu interpretieren: Eine "27" zu Beginn der Bezeichnung weist auf ein EPROM hin, welches nicht oder nur durch UV-Licht nach einer Programmierung wieder gelöscht und dann neu beschrieben werden kann. Wiederbeschreibbare EPROMs erkennt man an dem kleinen Fenster in der Bausteinmitte, worunter der Chip zu sehen ist. Um versehentliches Löschen zu verhindern, ist jedoch meist ein Aufkleber über diesem Fenster plaziert. PROMs ohne dieses Fenster sind nur ein einziges Mal zu programmieren und daher nicht wiederverwendbar.
Eine Weiterentwicklung sind elektrisch löschbare Bausteine. Eine "28" am Anfang der Typenbezeichnung weist auf einen löschbaren und wiederbeschreibbaren Typ hin. Diese Bausteine werden EEPROMs oder Flash-ROMs genannt. Sie sind auf fast allen neueren PC-Komponenten zu finden, die über ein eigenes BIOS oder Firmware verfügen. Sie werden dazu benutzt, bestimmte Konfigurationen eines Hardwareteils über die Stromabschaltung hinweg zu sichern. Meistens stellen die Komponenten selbst die benötigte Programmierspannung bereit. Es gibt unterschiedliche Typen von EEPROMs und Flash-ROMs, die sich ähnlich wie EPROMs durch ihre unterschiedlichen Programmierspannungen unterscheiden. Deshalb sind die Bausteine nicht in jedem Fall untereinander austauschbar.
Nach der "27" oder "28" folgt meistens ein "C"[17] oder "F"[18]. Danach wird die Größe des Bausteins in KiloBit angegeben. Anschließend folgt etwas angesetzt die mittlere Zugriffsgeschwindigkeit in Nanosekunden. In wenigen Fällen kann es mit älteren EPROMs wegen sehr langsamer Zugriffsgeschwindigkeiten zu Problemen kommen. Einige häufig auftretende Bezeichnungen sind in der Tabelle 9.3.2 zusammengefaßt.

Gelöscht werden können die klassischen EPROM's mit einer UV-Lampe[19]. Zum Programmieren benötigt man einen sogenannten EPROMMER oder

[15] Die Verwendung von Flash-Bausteinen für das Mainboard-BIOS kam erst mit Verwendung des PCI-Busses auf.

[16] Programmable Read Only Memory

[17] steht für CMOS, ein Chiptyp, der geringeren Strombedarf als TTL (TransistorTransistorLogic) hat

[18] weist auf ein Flash-ROM hin

[19] Durch das ultraviolette Licht werden nach einer Zeit von einer viertel bis halben Stunde die Chipstrukturen zurückgesetzt, so dass sie erneut elektrisch programmierbar

Chipaufdruck	Typ	Größe	Anmerkungen
2764-200	EPROM	8 kByte	alte Novell-Boot-ROMs
27128-200	EPROM	16 kByte	alte Mainboards
27C128-180	EPROM	16 kByte	meist als Pärchen installiert
27C256-180	EPROM	32 kByte	3-486er Boards
28C256-180	EEPROM	32 kByte	486er Boards
27C512-150	EPROM	32 kByte	spezielle 3-486er Boards
28F512-120	Flash-ROM	64 kByte	neuere Boards
28F010-100	Flash-ROM	128 kByte	neuere Boards
28F020-100	Flash-ROM	256 kByte	aktuelle Boards
28F040-100	Flash-ROM	512 kByte	neueste Boards

Tabelle 9.1: Bezeichnungen verfügbarer ROM-Bausteine

EPROM-Brenner. Rechenzentren, Universitätsinstitute der Physik oder Informatik verfügen normalerweise bereits über solche Geräte. Evtl. Neuanschaffungen sind als Kostenpunkt in der Gesamtbetrachtung abzuschätzen. Flash-ROMs und EEPROMs können ebenfalls ohne die Hilfe eines solchen Programmiergerätes mit Daten bestückt werden, da die benötigten Programmierspannungen nicht mehr sehr hoch sind und von den PC-Komponenten meistens bereitgestellt werden. Eine weitere Option besteht im Selbstbau einer Flash-Card. Hierzu sei auf das Netbootpaket verwiesen, welches im Anhang B.3 besprochen wird. Die Flash-Card kann in einem ISA-Slot Platz finden und auch dauerhaft anstelle des Boot-ROM auf der Netzwerkkarte eingesetzt werden.
Eine zusätzliche Möglichkeit für experimentelles Vorgehen besteht im Umprogrammieren des Mainboard-BIOS, wenn es sich in Flash-ROM-Bausteinen befindet. Eine ausführliche Anleitung zur Benutzung der DOS-Programme **cbrom.exe** und **amibcp.exe** findet sich im Anhang (Abschnitt A).
In den meisten Fällen ist die Boot-ROM-Option der Netzwerkkarte entweder über die Konfigurations- und Diagnosesoftware oder einen Jumper zu aktivieren. Weiterhin muss ein Adressbereich für das ROM ausgewählt werden. In diesem Speicherbereich sollten sich keine anderen BIOS-Extensions[20] befinden, was aber auf einem Terminal kaum vorkommen sollte.
Verfügt die Netzwerkkarte über keinen Sockel, so kann der Code auch über eine Diskette gebootet oder auch als Boot-Option z.B. unter Windows9X[21]

sind. Dieser Vorgang läßt sich chipabhängig bis zu 1000 Mal wiederholen

[20] z.B. eines SCSI-Controllers

[21] Die bereits vorliegenden Optionen, z.B. zum abgesicherten Start, lassen sich um eine Etherboot-Option erweitern und analog zum erstgenannten erreichen. Dazu kann man in den Bootsoftwarepaketen neben dem ROM-Code auch Code für die Floppy-Disk compilieren. Da diese Dateien sehr klein sind, bietet sich hier die Wiederverwertung alter 5,25"

eingetragen werden.
Eine weitere Variante besteht in der Nutzung älterer (und auch evtl. auf dem Netzwerkanschluß defekter) ISA-Netzwerk oder -Adapterkarten mit EPROM-Sockel. Diese Karten können zusätzlich in den Rechner eingebaut werden und liefern ausschließlich die Logik zum Erkennen und Ausführen des Boot-ROM's. Dazu muss auch hier die Boot-Option der Netzwerkkarte eingeschaltet werden, bzw. das installierte EPROM gegen das gewünschte Boot-ROM getauscht werden. Eine Voraussetzung hierfür besteht jedoch in der Unterstützung der gewünschten EPROM-Größe, welche bei älteren Karten teilweise nicht gegeben. Dieser Vorgang funktioniert deshalb, da der ISA-Bus über keine Erkennungsmöglichkeiten, wie PCI- und Vendor-ID's verfügt und deshalb Code für komplett andere Adapter ausführen kann.

9.3.3 Dual-Boot über Windows-Bootloader

Soll ein Computer mit verschiedenen Betriebssystemen genutzt werden, z.B. neben dem auf Festplatte installierten Microsoft Windows NT oder 2000 das Gerät als X-Terminal oder Diskless X-Station starten, verwendet man bei Etherboot dazu die Compile-Option -DASK_BOOT[22]. In einigen Fällen gibt es jedoch Hardwarekonflikte zwischen Windows-NT und dem installierten EPROM. Für diese Situation und zu Testzwecken lassen sich unter DOS ausführbare Dateien produzieren (z.B. mit "make rtl8139.com"), welche die *.rom Images in ihrer Funktionalität um die notwendigen Eigenschaften ergänzen.
Soweit man nicht auf einen bereits vorhandenen DOS-Bootsektor zurückgreifen kann, erzeugt man ihn mittels Formatieren und Installieren einer Festplatte mit DOS, bevor man NT einspielt (vgl. Win-NT Multiboot-HOWTO). Man benötigt zudem die DOS Systemdateien (*io.sys*, *msdos.sys* oder *kernel.sys* wenn man FreeDOS verwendet) und kopiert diese in das Verzeichnis des NT-Loaders. Wenn man die **.com* Datei über die *autoexec.bat* aufrufen möchte, sollte man auch die entsprechende *command.com* bereitstellen oder man nennt die Etherbootdatei einfach nach *command.com* um. Diese wird dann anstelle der DOS-Shell aufgerufen, was Benutzereingriffe, wie Abbrüche etc. verhindert.
Anschließend wird in der *boot.ini* eine Zeile hinzugefügt, als solle DOS gebootet werden:

```
[boot loader]
```

Laufwerke geradezu an. Die Bootdatei bekommt einen minimalen Loader für den Bootsector der Floppy-Disk davorgehängt und wird dann direkt auf die Diskette geschrieben ("make name_der_netzwerkkarte.fd0").

[22] weitere Optionen finden sich im Anhang B.2

```
timeout=20
default=C:\bootsect.dos
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Win NT Workstation"
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Win NT Workstation,
[VGA-Modus]" /basevideo /sos
C:\bootsect.dos="DHCP/TFTP (Linux diskless via etherboot)"
```

Unter Windows 95 bzw. 98 kann die ***.com**-Datei einfach über das Bootmenü aufgerufen werden.
Der Windows2000-Loader erlaubt ein einfacheres Vorgehen:

```
[boot loader]
timeout=20
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Win 2000 Prof."
C:\="Linux via Etherboot-.COM-File booten"
```

Voraussetzung hierfür sind die Existenz der Windowsstartdateien **io.sys**, **msdos.sys** und der Kommandointerpreter **command.com**. Über die **autoexec.bat** wird dann einfach die Etherboot-***.com**-Datei angestartet.

9.3.4 Dual-Boot mittels Linux Loader

Für den Linux Loader (LILO) gibt es ein Etherbootpatch, der es ermöglicht Etherboot so zu kompilieren, dass es direkt am LILO-Prompt aufgerufen werden kann. Hierfür sind die Linux-Kernelquellen nicht notwendig. Für nähere Ausführungen sei auf die umfangreiche Dokumentation von LILO und Etherboot verwiesen.

Kapitel 10

Einrichtung der Software der Clients

10.1 Überblick

Nachdem die Bootsoftware auf dem Thin-Client eingerichtet wurde, erfolgt die Konfiguration auf der Seite der Server, da nur hier alle Einstellungen dauerhaft gespeichert sind. Deshalb beschäftigen sich die nächsten Abschnitte mit der Generierung eines speziellen Linuxkernels, welcher für den festplattenlosen Bestrieb angepaßt wird. Ein gewisses Augenmerk ist auf den Bereich des Filesystems zu richten, in dem die Kernelschnittstellen[1] zu einzelnen Geräten liegen.
Gegenüber dem Betrieb einer klassischen Linuxworkstation sind während des Bootvorganges einige Besonderheiten zu beachten, welche in einem eigenen Abschnitt abgehandelt werden. Dieses Kapitel arbeitet den Bootvorgang der Reihenfolge nach ab, wobei mit der Erstellung des Kernels begonnen wird, es dann um die Anpassung dieses an die Bootsoftware Etherboot geht ehe anschließend die verschiedenen Bootskripten vorgestellt werden. Die Konfiguration der Grafikhardware, die Hardwareabhängigkeiten außerhalb des Kernels abhandelt, wird abschließend betrachtet.

10.2 Erstellen des Netkernels

Damit der Linux-Kernel in der beschriebenen Umgebung aus dem Netz gebootet werden kann, sind einige spezielle Einstellungen erforderlich: Die

[1] Üblicherweise */dev*

Unterstützung des Ramfilesystems bzw. der Ramdisk und ihres Filesystems, des NFS als Rootfilesystem, sowie der entsprechenden Netzwerkkarte (hier ist die populäre RTL8139 basierte 100 Mbit-Karte als Beispiel eingetragen) können statisch in den Kernel eingebunden sein. Eine andere Herangehensweise liegt in der Verwendung einer "Initial Ramdisk", was zwar einen höheren Aufwand beim Erstellen des Bootimages bedeutet, jedoch den Kernelumfang zusätzlich reduzieren kann. Weiterhin ist das Flag für die automatische IP-Konfiguration zu setzen. Der Ausschnitt aus der Konfigurationsdatei des Kernels (ab Version 2.4.X) sieht wie folgt aus:

```
[...]
CONFIG_PACKET=y   # Grundlage
CONFIG_FILTER=y   # für
CONFIG_UNIX=y     # dhclient
CONFIG_INET=y     #
[...]
CONFIG_IP_PNP=y   # Kernel übernimmt IP-Konfiguration
[...]
CONFIG_RTL8139=y  # Unterstützung für Netzwerkkarte
[...]
CONFIG_RAMFS=y    # "Filesystem" der Ramdisk
CONFIG_NFS_FS=y   # NFS ist die Basis unseres Rootfilesystems
CONFIG_ROOT_NFS=y # Das Rootfilesystem wird per NFS erwartet
[...]
```

Nicht ganz so flexibel erweist sich der Einsatz einer Ramdisk anstelle des Ramfilesystems[2]. Diese hat eine feste Größe, die beim Kompilieren des Kernels angegeben wird. Um die Ramdisk auch benutzen zu können, muss man sie mit einem Filesystem versehen. Hier bietet sich das Minix-Filesystem an, da es einen sehr geringen Overhead für Dateiinformationen benötigt. Dafür besteht die Beschränkung in der Länge der Dateinamen auf maximal 32 Zeichen. Die Konfiguration[3] für einen 2.2er Kernel sieht (in Ausschnitten) so aus:

```
[...]
CONFIG_BLK_DEV_RAM=y # Erstellen einer Ramdisk
CONFIG_BLK_DEV_RAM_SIZE=4096 # Definieren der Ramdiskgröße
[...]
CONFIG_PACKET=y   # Grundlage
CONFIG_FILTER=y   # für
```

[2] Das Ramfilesystem steht erst mit Kernel 2.4.X zur Verfügung
[3] gespeichert in der Datei *.config*

```
CONFIG_UNIX=y      # dhclient
CONFIG_INET=y      #
[...]
CONFIG_IP_PNP=y    # Kernel übernimmt IP-Konfiguration
[...]
CONFIG_RTL8139=y   # Unterstuetzung fuer die
                   # entsprechende Netzwerkkarte
[...]
CONFIG_MINIX_FS=y # Minix-FS verwenden
CONFIG_NFS_FS=y    # NFS ist die Basis unseres Rootfilesystems
CONFIG_ROOT_NFS=y # Das Rootfilesystem wird per NFS erwartet
[...]
```

Die Ramdisk für X-Terminals reicht mit 512 kByte aus, für Diskless X-Stations sollten mindestens 4 MByte, in Abhängigkeit der verwendeten Applikationen und des Hauptspeichers evtl. auch mehr vorgesehen werden.
Viele andere Optionen (wie zum Beispiel die Unterstützung serieller Mäuse oder des Agpgart für einige Grafikkarten) können später als Module nachgeladen werden. Zum Schluss muss das neu generierte *bzImage* mit einer speziellen Bootsignatur versehen werden: **mknbi-linux –outputfile /nfsroot/xtbootimg bzImage**. Das Tool **mknbi-linux** ist Bestandteil des Etherbootpakets (siehe hierzu den Anhang B.4).
In einigen Fällen ist die realisierbare Länge der Kernelkommandozeile[4] anzupassen, da auf diesem Wege neben der vollständigen IP-Konfiguration weitere Parameter übermittelt werden. Dazu können z.B. die VGA-Auflösung für das Framebufferdevice[5], die Umleitung der Konsolenausgabe und Parameter für einzelne Module zählen.

10.3 Das Devicefilesystem

Das "devfs" ist eine Weiterentwicklung des devpts, des Virtual-Terminal-Filesystems. Es bildet alle im Rechner vorhandenen Devices in einer eigenen logischen Verzeichnisstruktur unterhalb von */dev* ab. Die Einträge werden dynamisch anhand der Liste der beim Booten des Kernels gefundenen Geräte erzeugt. Leider sind noch nicht alle Devices implementiert und beim Laden von Kernelmodulen funktioniert der Mechanismus nicht sicher. Der Bereich des "devpts" bildet eine Untermenge des "devfs" für die virtuellen

[4] Hierzu zählen die Zahl der übergebbaren Parameter in der Datei *linux/init/main.c* und die Anzahl der Zeichen in der Datei *linux/arch/i386/kernel/setup.c*.

[5] Grafikausgabe oder hochauflösender Text in die klassische Konsole; Voraussetzung ist jedoch immer die Unterstützung durch die Hardware

Konsolen und muss nicht mehr separat eingebunden werden. Ein Hintergrundprgramm, in diesem Fall der **devfsd** sorgt für Kompatibilitätslinks nach klassischem Muster im neuen Devicefilesystem.
Ein Vorteil des "devfs" liegt darin, dass nicht mehr alle vorkommenden Devices im vorneherein definiert werden müssen, sondern dass nur alle benötigten automatisch angelegt werden. So läßt sich das NFS-Problem im Zusammenhang mit den klassischen Deviceeinträgen umgehen, welches in Abschnitt 8.4 angedeutet wurde.

10.4 Der Bootvorgang

10.4.1 Überblick

Der Bootvorgang eines Diskless-Clients weicht von dem einer klassischen Linux Workstation gerade zu Beginn entscheidend ab. Da auf dem Client keinerlei Konfigurationsdateien vorliegen, müssen sämtliche dieser Daten aus dem Netz beschafft werden. Die IP-Konfiguration liegt jedoch, anders als bei der klassischen Workstation, bereits mit dem Start des Kernels vor und muss nicht mehr über die Runlevel-Skripten[6] abgehandelt werden.
Im Gegensatz zum klassischen Start mittels Bootloader[7] muss der Kernel zuerst übers Netz kopiert und anschließend von der Bootsoftware gestartet werden. Dieser Prozess wird im nächsten Absatz beschrieben. Die Anforderungen an das *boot*-Skript und die Runlevel-Skripten liegen etwas anders als gewohnt und werden in den darauffolgenden Abschnitten erläutert. Dabei geht es dann auch um das Erzeugen der notwendigen Konfiguraionsdateien und Setzen bestimmter Parameter.

10.4.2 Booten des Netkernels

Der Netkernel wird nach der DHCP-Anfrage durch den Bootcode auf der Netzwerkkarte oder im Flash-ROM des Mainboards per TFTP bzw. NFS übertragen und anschließend mit den entsprechenden Netzwerkparametern gestartet. Diese werden dem Kernel per Optionszeile übermittelt, welche durch **mknbi(-linux)** oder durch DCHP-Optionen modifiziert werden kann. Sobald der Kernel gestartet wird, gibt die Bootsoftware die Kontrolle über den Rechner ab und beendet sich[8]. Nur wenn eine gültige IP-Konfiguration

[6] Die meisten Linuxdistributionen, wie das hier verwendete SuSE-Linux setzen auf die Initialisierung und Steuerung der Maschine im Stil des System-V-Init.

[7] üblicherweise **lilo**, **lodalin**, **grub** ...

[8] Nun spielt auch der geladene Netzwerkkartentrieber keine Rolle mehr, da der Kernel diese Funktion übernehmen muss

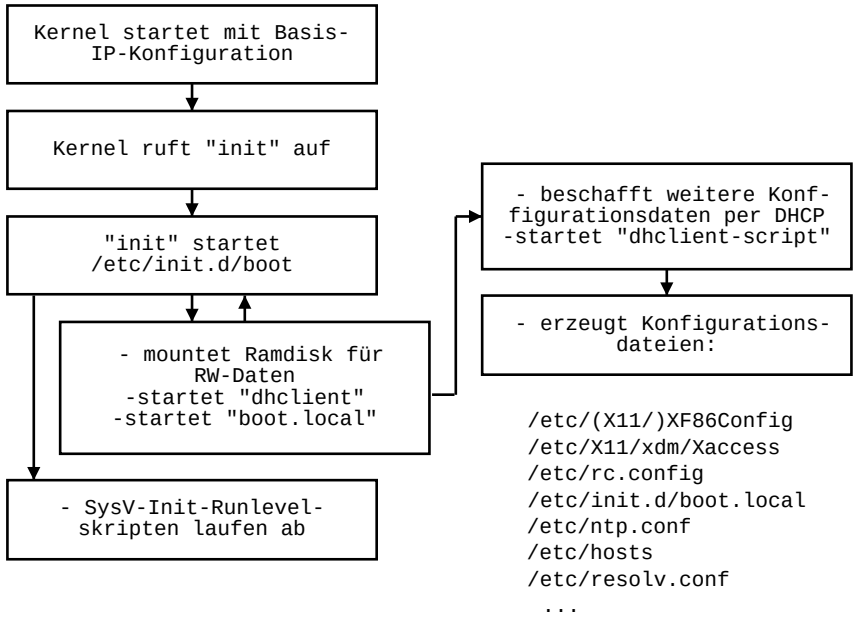


Abbildung 10.1: Boot- und Konfigurationsvorgang

übertragen, die korrekte Netzwerkkarte in den Kernel kompiliert und die IP-Autokonfiguration eingeschaltet wurde, kann der Rechner nach Hochlauf des Kernels im Netzwerk erreicht werden. Im nächsten Schritt wird das Rootfilesystem per NFS gemountet und der Init-Prozess gestartet.

10.4.3 Die Startskripten

Die Software-Umgebung der Thin-Clients lehnt sich an die SuSE-Linuxdistribution an, wobei entsprechende Anpassungen in *(/nfsroot/dxs)/etc/init.d* vorgenommen sind. Diese Einstellungen sind auf alle Distributionen, welche das System-V-Init verwenden, leicht übertragbar. Das Start-Skript **boot** welches von **initd** als erstes aufgerufen wird, wurde für eine festplattenlose Umgebung umgeschrieben, andere Skripten, z.B. zur Initialisierung des Netzwerkes, entfallen. Alle verwendeten und angepaßten Skripten sind im Anhang (Abschnitt D) vollständig abgedruckt.
Als erster Vorgang nach dem Transfer und Start des Netkernels wird mittels des *(/nfsroot/dxs)-/etc/init.d/boot*-Skriptes das Ramfile oder die Ramdisk[9]

[9]In diesem Fall muss noch eine Formatierung erfolgen.

ReadWrite gemountet (Der entsprechende Eintrag in der *(/nfsroot/dxs) /etc/fstab* muss vorhanden sein). Anschließend wird aus dem gemounteten Rootfilesystem (hier *(/nfsroot/dxs)/var/ram*) der Teil des Dateisystems, der ReadWrite-Zugriffe gestatten soll, in das Ramfilsystem entpackt. Unterhalb dieses Verzeichnisses wird der Inhalt der zukünftigen Ramdisk abgebildet, so dass er bei Bedarf geändert und immer wieder neu generiert[10] werden kann.
Die entsprechende **tar**-Option hält den Dateibesitzer und dessen Zugriffsrechte bei. Diese Bereiche umfassen einen Teil des *(/nfsroot/dxs)/var* Verzeichnisses (z.B. *run* für das Schreiben des *utmp*-Files), *(/nfsroot/dxs)/tmp* für die XFree86-Sockets und Locks, den Tastaturkompiler des XFree86 etc. Dateien im übrigen Verzeichnisbaum, auf die Schreibzugriffe notwendig sind (*(/nfsroot/dxs)-/etc/resolv.conf, (/nfsroot/dxs)/etc/X11/XF86Config, (/nfsroot/dxs)/etc/rc.config ...*) werden geeignet in das Ramfile respektive die Ramdisk mit symbolischen Links eingebunden.

```
[...]
. /etc/rc.config.default
echo "Running $0."
echo -n "Mounting local file systems..."
mount -van || ECHO_RETURN=$rc_failed
echo -e $rc_done
tar  -xpzf /var/ram/ramdisk.tgz -C /RAM
echo -e $rc_done
[...]
```

Anschließend werden die Nameservice- und andere Netzwerkparameter, wie X- und Timeserver über den ISC dhclient, das **dhcpd** Gegenstück, eingetragen und konfiguriert[11]. Dies geschieht mittels **dhclient-script**, welches über das Kommando **dhclient** im **boot**-Skript aufgerufen wird. Die *(/nfsroot/dxs)/etc/rc.config*, *(/nfsroot/dxs)/etc/crontab* und *(/nfsroot/dxs)/etc/X11/XF86Config* werden aus Defaultdateien[12] generiert.

```
[...]
echo -n "Setting up X-Terminal configuration ..."
ECHO_RETURN=$rc_done
/sbin/dhclient -q eth0 -s $SERVER >/dev/null 2>&1 \
     || ECHO_RETURN=$rc_failed
echo -e "$ECHO_RETURN"
```

[10] mittels **tar -cpzf ramdisk.tgz *** in diesem Verzeichnis

[11] Eine genaue Beschreibung dieses Tools erfolgt im nächsten Abschnitt.

[12] z.B. *(/nfsroot/dxs)/etc(/X11)/X86Config.default* für die XFree86 Einstellungen und (/nfsroot/dxs)/etc/rc.config.default für die zentrale SuSE-Konfiguationsdatei.

```
[...]
```

Werden bestimmte Module, wie der *agpgart.o* oder Treiber für die Soundkarte benötigt, können diese entweder dynamisch durch den Kernelmodullader oder über das Bootskript für Benutzererweiterungen *(/nfsroot/dxs)/etc /init.d/boot.local* initialisiert werden. Die Modularisierung spart Kerneloverhead und ermöglicht die Verwendung unterschiedlicher Hardware auf Basis eines einzigen gemeinsamen Filesystems. Die Verwendung von Modulen statt des festen Einbindens der Treiber beschleunigt den Startvorgang des Thin-Clients, da das üblicherweise ausgeführte Autoprobing vor dem Aktivieren des Treibers nur bei Bedarf erfolgt.
Der lokale X-Server des Terminals kann zum Anbieten eines grafischen Logins einen speziellen Host anfragen (-query), Broadcasts absetzen oder über den eigenen Chooser eine Liste erreichbarer X-Login-Server zur Auswahl stellen. Diese Liste und der Typ der Anfrage wird wiederum über ein DHCP-Optionenfeld übermittelt. Weitere Optionen steuern über die Modifikation der *(/nfsroot/dxs)/etc/rc.config*, ob bestimmte Dienste, wie der **snmpd** oder **cron** gestartet werden. Beide Dateien liegen als *.default im */etc*-Baum und werden durch **dhclient-script** geparst und als XF86Config bzw. *rc.config* in der Ramdisk abgelegt. Nach dem Durchlauf dieser beiden Basisskripten gibt es kaum Unterschiede zu einer klassischen Linux-Workstation. Die meisten Start-/Stop-Skripten können direkt übernommen werden.

10.5 "dhclient" Einstellungen

Die Startsoftware auf dem Boot-ROM bezieht vom Server nur die nötigsten Einstellungen, um eine vollständige IP-Konfiguration generieren zu können. Die Daten werden dem Kernel direkt übergeben. **dhclient** und **dhclient-script** sind das Herzstück der Konfiguration. Dem Kernel werden zwar die Basisnetzwerkparameter mitgeteilt, alle weiteren Daten müssen jedoch in einem zweiten Schritt angefordert werden. Der Umfang aller möglichen Kernelkommandoparameter kann dem "BootPrompt-HOWTO" entnommen werden, welches üblicherweise jeder Linuxdistribution beiliegt oder nachinstalliert werden kann.
Auf der Terminalseite sorgt in einem zweiten Schritt **dhclient** für die Konfiguration der einzelnen Variablen, indem die Daten vom Server angefordert und durch **dhclient-script** in die Konfiguration der DXS eingefügt werden. Die Konfiguration, welche Daten überhaupt angefordert werden sollen, bzw. für den Betrieb zwingend notwendig sind oder nur bei Vorhandensein abgefragt werden, erfolgt in der Datei */etc/dhclient.conf*.
Die Trennung des Tools zur Beschaffung der Daten (durch das ausführbare

Programm **dhclient**) und die letztendliche Konfiguration der unterschiedlichen Aspekte durch ein Shellskript erlaubt eine weitgehende Anpassung an die jeweiligen Gegebenheiten. So muss nicht für jede Änderung bzw. jedes neue Datenfeld das ausführbare Programm neu übersetzt werden, das Debugging wird einfacher und die Portabilität auf andere Plattformen leichter. Nach der Spezifikation des DHCPv3 lassen sich weitere Optionen definieren[13], über die dann hardwarespezifische Daten und Anpassungen übermittelt werden können. So ist es denkbar, alle hardware- und softwarespezifischen Konfigurationen in einer Datei zusammenzufassen. Diese umfaßt zuallererst die bereits vordefinierten Felder zu verschiedenen Netzdiensten, wie Print-, Log-, XDMCP-, Timeserver. Darüberhinaus ist es möglich sogenannte "Vendoroptions", d.h. über die fest definierten Standardoptionen hinaus, eigene Optionen zu definieren und mittels dieser weitere Informationsblöcke zu übertragen. Im vorgestellten Fall sind dieses die Einstellungen zum Start bestimmter Dienste, wie z.B. **snmpd** oder **sshd**. Darüberhinaus werden Strings zur Konfiguration des Grafikservers (siehe hierzu Abschnitt 10.6) übertragen.
Benutzerdefinierte Optionen werden analog zu den vordefinierten abgefragt. Das zu übertragende Datenmaximum wird durch die MTU-Size[14] des Netzwerkes begrenzt. Wie auch bei den Servereinstellungen gilt: Soll viel Information übertragen werden, muss die Größe des Bootreplypakets angepaßt werden. Dieses läßt sich vom Server mit "send dhcp-max-message-size INT"[15] anfordern. Eine weitere, zeitlich jedoch weit aufwändigere Methode, ist eine kaskadierte DHCP-Anfrage, bei der die Optionen auf zwei Anfragen verteilt werden.
Im Normalfall sehen die wichtigsten Einträge der */etc/dhclient.conf* so aus:

```
[...]
send dhcp-max-message-size 1024;
[...]
request routers, domain-name, domain-name-servers,
        host-name, font-servers, ntp-servers,
        x-display-manager,lpr-servers, start-sshd,
        start-dnetc, start-lpd, start-rwhod, start-cron,
        crontab-entries;
[...]
```

Diese Daten sollten mit den entsprechenden Einträgen der *dhcpd.conf* korrespondieren. Es ist jedoch genauso vorstellbar, für bestimmte Werte Defaults

[13] Diese werden wie in der **/etc/dhcpd.conf**, eingetragen und angesprochen.
[14] Maximum Transfer Unit
[15] Die maximale Paketgröße bestimmt sich aus der MTU-Size abzüglich IP- und UDP-Header.

zu definieren, die eingesetzt werden, wenn keine Daten vom Server bezogen werden konnten:

```
[...]
default x-server-defs   "de IMPS/2 psaux 96 150 1024x768 nv 16";
default start-x         "indirect";
default start-snmp      "yes";
default start-sshd      "yes";
default start-dnetc     "yes";
default start-xdmcp     "yes";
default start-cron      "yes";
default crontab-entries "";
default start-lpd       "yes";
[...]
```

Durch das Setzen der Default-Parameter bereitet es kein Problem, das Erzeugen der verschiedenen Konfigurationsdateien dem **dhclient-script** zu überlassen. **dhclient-script** ist ein umfangreiches Shell-Skript, das direkt die erforderlichen Konfigurationsdateien manipuliert oder bestimmte Systemvariable direkt setzt. Dieses Skript läßt sich in seiner Funktion mit den Runlevel-Skripten einer Standard-Linux-Installation vergleichen. Die Verteilung der Aufgaben zur Systemkonfiguration hängt von den Präferenzen des Systemadministrators ab: Möchte man eine recht hohe Ähnlichkeit zur Defaultinstallation einer bestimmten Linuxdistribution beibehalten, um beim Update nur wenige Abhängigkeiten beachten zu müssen, oder die Konfiguration direkt dem **dhclient-script** überlassen.
Das vollständige Skript kann dem Anhang (siehe Abschnitt D.2.2) entnommen werden. Mit dem ISC-DHCP wird ein Beispielskript verteilt, welches anschließend um alle Optionen erweitert werden muss, die über die Minimalkonfiguration von IP-Adresse, Hostname und DNS-Auflösung hinausgehen. Alle Konfigurationsdateien, die von diesem Vorgang berührt werden, sind deshalb in die Ramdisk verlinkt. So kann das Basisfilesystem ReadOnly bleiben und einheitlich für alle Maschinen verwendet werden. Hierin liegt eine zentrale Komponente der Vereinfachung: Bei einer sehr hohen Anzahl von Thin-Clients muss nicht mehr für jede Maschine ein eigener Dateibaum gepflegt werden. Deshalb wird man versuchen, alle Unterschiede zwischen der Hardware- und Software-Installation für alle hostspezifischen Daten mittels DCHP und Ramdisk bzw. Ramfile abzubilden.

10.6 X11-Konfiguration

Ziel dieses Installationsverfahrens ist es, ein gemeinsames Filesystem für alle betriebenen X-Terminals bzw. Diskless X-Stations zur Verfügung zu stellen. Bei einer steigenden Anzahl von Clients wird es sehr schnell unübersichtlich, für jedes Gerät ein eigenes Filesystem anzulegen. Updates würden im Aufwand gewaltig steigen, der Platzbedarf wäre nicht unerheblich. Dieses gemeinsame Filesystem umfaßt deshalb alle statischen Daten, welche zentral ausschließlich lesbar zur Verfügung gestellt werden. Alle Bereiche, welche gesonderte Anpassungen an den einzelnen Client erordern, müssen im lokalen Filesystem (Ramdisk mit Minix oder Ramfile) angelegt werden. Das stellt besondere Anforderung an die Konfiguration der Grafikkarte für XFree 3.3.6 oder 4.X.Y. Die Konfiguration der *XF86Config* soll an dieser Stelle exemplarisch etwas ausführlicher beschrieben werden, solche oder ähnliche Vorgehensweisen bieten sich auch für andere Einstellungsdateien an, die für den Betrieb der Thin-Clients generiert werden müssen.
Freundlicherweise erleichtert der Aufbau der XFree86 Konfigurationsdatei diese Aufgabe. So kann man sich vorstellen, für eine Auflösung verschiedene Modelines[16] gleichzeitig anzugeben. Davon wird jeweils die mit den am besten geeigneten bzw. höchsten Bildwiederholraten ausgewählt, die zu einer gebenen Obergrenze[17] der Vertikal- und Horizontalfrequenz paßt. Bei fast allen neueren Grafikkarten funktioniert das Autoprobing auf die Größe des Grafikspeichers ohne Probleme. Weiterhin wird die maximale Bildschirmauflösung angegeben, um zu verhindern, dass Monitore mit guten physikalischen Eigenschaften eine für den Benutzer zu hohe Auflösung anbieten.
In einer bestimmten Farbtiefe wird wiederum immer die höchstmögliche Bildauflösung gewählt, so dass sich durch Angabe der gewünschten Farbtiefe und der Monitorfrequenzen folgendes Einstellungen ergeben:
Je nach eingesetzter Monitorklasse kann es sinnvoll erscheinen auch höhere Auflösungen anzugeben. Die in der Tabelle zuerst aufgeführten Auflösungen sind eher inakzeptabel für moderne Arbeitsplätze, zeigen aber, dass sich auch ältere Hardware in einem gewissen Rahmen weiterverwenden läßt. Die Maximalauflösung auf 640x480 zu setzen, fällt wegen des zum Arbeiten zu kleinen Desktops weg.

XFree 3.3.6 Da einige Grafikhardware[18] nicht mehr vom neuen XFree86 unterstützt werden wird, bleibt für diese Grafikkarten die Verwendung der

[16] Beschreibungdaten zur Ansteuerung des Monitors

[17] Diese ist bei einigen älteren Geräten zwingend einzuhalten, da eine Übersteuerung den Monitor beschädigen könnte.

[18] große Teile der S3-Linie, ältere ISA-Karten, einige ATI-Karten, viele ältere Grafikkarten nur in 8 bit Farbtiefe

Grafik-RAM	Farbtiefe	Horiz.freq.	Vert.freq.	Auflösung
1024 kByte	8 bit	36 kHz	83 Hz	1024x768 interlaced
1024 kByte	16 bit	36 kHz	70 Hz	800x600
2048 kByte	16 bit	36 kHz	83 Hz	1024x768 interlaced
2048 kByte	16 bit	56 kHz	72 Hz	1024x768
2048 kByte	16 bit	60 kHz	75 Hz	1024x768
2048 kByte	16 bit	65 kHz	80 Hz	1024x768
4096 kByte	16 bit	96 kHz	80 Hz	1280x1024
4096 kByte	24 bit	65 kHz	80 Hz	1024x768
8192 kByte	16 bit	96 kHz	80 Hz	1600x1200
8192 kByte	24 bit	96 kHz	80 Hz	1280x1024

Tabelle 10.1: Verhältnis des Grafikspeichers zu Farbtiefe und Auflösung

Version 3.3.6. Jedoch stehen dann einige Funktionen, wie "AntiAliasing" und "TrueType"-Fonts nicht oder nur eingeschränkt zur Verfügung.
Die für das Terminal zu verwendende *(/nfsroot/dxs)/etc/XF86Config* wird aus einer Defaultdatei[19] durch parsen mittels **sed** erzeugt. Dies geschieht durch den Aufruf von **dhclient-script**, welches im Anhang D.2.2 näher erläutert wird. Bei XFree86 Version 3 ist es darüberhinaus noch notwendig, den Link auf den korrekten Grafikserver[20] zu erzeugen. Beim Parsen der Datei und Anlegen der Konfiguration in der Ramdisk[21] wird zuerst in der "Section Keyboard", "LANG" durch die entsprechende angeschlossene Tastatur ersetzt.

```
Section "Keyboard"
    Protocol    "Standard"
    AutoRepeat  500 5
    XkbRules    "xfree86"
    XkbLayout   "LANG"
    XkbModel    "pc102"
    XkbVariant  "nodeadkeys"
EndSection
```

Im nächsten Abschnitt: "Section Pointer" wird das Mausprotokoll ("MP") durch das Protokoll des angeschlossenen Maustypes ersetzt. Die Werte ergeben sich aus den unter XFree86 denkbaren Einstellungen[22]. Die Emulation

[19] Hier: /etc/XF86Config.default
[20] XFree86_SVGA, XFree86_S3, XFree86_S3V, XFree86_Mach64, ...
[21] */etc/XF86Config* ist ein Link auf */RAM/etc/XF86Config*
[22] MouseMan, MicroSoft, PS/2, IMPS/2 ...

der 3-Tasten-Maus wird standardmäßig gewählt, da diese Einstellung bei "richtigen" 3-Tasten-Mäusen nicht stört.

```
Section "Pointer"
    Protocol    "MP"
    Device      "/dev/MD"
    Emulate3Buttons
EndSection
```

Im nächsten Konfigurationsabschnitt der Datei "Section Monitor" werden die horizontale Synchronisationsfrequenz ("HS") und vertikale Wiederholrate ("VS") eingetragen. Die darauf folgenden Modelines ergeben sich aus dem Spektrum der eingesetzten Monitore und sind im Fall neuerer Monitore oder evtl. moderner TFT-Displays zu erweitern.

```
Section "Monitor"
    Identifier "Default"
    Vendorname "Standard"
    HorizSync   31.5 - HS
    VertRefresh 50 - VS
[...]
EndSection
```

Unter XFree 3.3.6 können folgende im Wesentlichen zwei Typen[23] - "accel" und "svga" - angegeben werden. Es muss jedoch darauf geachtet werden, dass alle verwendeten Grafikkarten durch entsprechend installierte X-Server unterstützt werden. Hierbei liegt das Augenmerk auf der geeigneten Verlinkung des Serverbinaries, welches im Gegensatz zur neueren Version hardwareabhängig zu wählen ist. (Diese Einstellungen muss das zentrale Konfigurationsskript **dhclient-script** vornehmen.)

```
Section "Screen"
    Driver      "DRV"
    Device      "Graphics"
    Monitor     "Default"
    DefaultColorDepth CDP
[...]
```

Eine letzte Eintragung erfolgt realtiv am Ende der Datei, wenn das Kürzel "CDP" durch die gewünschte Farbtiefe im Abschnitt "Section Screen" ersetzt wird. Kommt es hier zu einer Vertauschung von Daten, kann es passieren, dass der Grafikserver nicht gestartet wird.

[23] Die Typen "vga", "hercules", "vga16" machen nur noch wenig Sinn

XFree 4.X.Y Die Konfiguration für XFree86 Version 4 ist leichter, da in jedem Fall immer das identische Binary (**(/nfsroot/dxs)/usr/X11R6/bin/XFree86**) aufgerufen wird und somit die dynamische Erzeugung des passenden Links mit **dhclient-script** entfällt. Die *XF86Config* unterscheidet sich etwas im Aufbau der letztvorgestellten.
Zu Beginn wird jedoch wiederum die Tastatur eingestellt und anschließend die Maus konfiguriert. Die Bezeichnung der Variablen unterscheidet sich nicht, wenn auch einige Mausprotokolle hinzugekommen sind.

```
Section "InputDevice"
    Identifier  "Keyboard1"
    Driver      "keyboard"
    Option      "XkbRules"      "xfree86"
    Option      "XkbLayout"     "LANG"
    Option      "XkbModel"      "pc102"
    Option      "XkbVariant"    "nodeadkeys"
EndSection
Section "InputDevice"
    Identifier  "Mouse1"
    Driver      "mouse"
    Option      "Protocol"      "MP"
    Option      "Device"        "/dev/MD"
    Option      "Emulate3Buttons"
EndSection
```

Die Monitorkonfiguration kann direkt von XFree 3.3.6 übernommen werden. Werden jedoch keine Modelines angegeben, versucht XFree Version 4 diese vom Monitor per Autodetect zu beziehen.
Für die Auswahl des Treibers für die verwendete Grafikhardware muss die "Section Device" angepaßt werden:

```
Section "Device"
    Identifier  "StdGraphics"
    Option      "power_saver"
    Driver      "DRV"
EndSection
```

Zum Dateiende hin erfolgt, wie bereits bekannt, die Einstellung zur Default-Farbtiefe. Diese kann auch direkt von XFree 3.3.6 übernommen werden. In diesem letzten Abschnitt werden alle Bildschirmauflösungen angegeben, die unter einer bestimmten Farbtiefe zur Verfügung stehen sollen. Die tatsächliche Auflösung ergibt sich entsprechend der Tabelle oben.

Kapitel 11

Besonderheiten der DXS

11.1 Unterschiede zu X-Terminals

X-Terminals stellen geringe Anforderungen an die Leistungsfähigkeit der eingesetzten Hardware und bieten sich deshalb vorwiegend im Einsatz älterer Gerätegenerationen an. Mit der ständigen Weiterentwicklung des Marktes und preiswerter werdenden PC-Systemen besteht nun auch die Möglichkeit, bereits die Einstiegsklasse heutiger Rechner für den direkten Benutzerbetrieb einzusetzen, um den Server zu entlasten. Voraussetzung hierfür sind eine bessere Speicherausstattung, die je nach Anforderung zwischen 96 und 256 MB pro Client liegt, sowie eine Integration in ein 100 Mbit-Netzwerk. Beim derzeigen Preisverfall der aktiven Netzwerkkomponenten und Speicherbausteine sollte dies kein Hindernis beim Aufbau moderner Arbeitstationen darstellen. Dieses ist zumeist preiswerter zur erreichen, als die Investition in entsprechend stärkere Serverplattformen, die üblicherweise nicht mehr am preiswerten PC-Markt zu realisieren sind.
Diskless X-Stations beziehen über das Rootfilesystem hinaus weitere Filesystembereiche vom Server. Die Antwortgeschwindigkeit bei Benutzerinteraktionen setzt sich aus der Rechner- und der Netzwerkgeschwindigkeit zusammen. Die Möglichkeiten der schnellen Grafikausgabe und die Benutzung der 3D-Schnittstelle unterstützt der Gerätetyp der X-Terminals nur rudimentär. Mit der Rückverlagerung der Useraktivitäten auf den lokalen Rechner stehen auf Diskless X-Stations nun wieder die vollen Bandbreiten des jeweiligen Bussystems der Maschine zur Verfügung. Mit der Realisierung der Anbindung der NFS-Server an das evtl. bereits existierende Gigabit-Backbone-Netz lassen sich Antwortgeschwindigkeiten des Netzwerkfilesystems nahe derer lokaler Festplatten erreichen. Nutzt man weiterhin die Caching-Möglichkeiten des Kernels für häufig angeforderte Da-

tenblöcke durch großzügigen Ausbau des Hauptspeichers, lassen sich die Unterschiede noch stärker reduzieren, ohne die Kostenbilanz zu Ungunsten der Thin-Clients zu verschieben.

11.2 Filesystemüberlegungen

Im hier gezeigten Fall werden wärend des Startvorganges[1] der DXS weitere Bereiche des Serverfilesystems hinzugemountet. Dazu gehören */usr*, */opt* und bei Bedarf */tmp/users*. Die beiden erstgenannten Bereiche enthalten alle wichtigen Applikationen, die Dokumentations- und Shareverzeichnisse. Voraussetzung für den Zugriff auf die Programme und Bibliotheken im Serverfilesystem ist eine identische Prozessorarchitektur auf der Client-Seite. Die Programme und Bibliotheken werden ReadOnly eingemountet, um keine Sicherheitsprobleme auf dem Server entstehen zu lassen. Eine Freigabe des Rootfilesystems mit der "no_root_squash"-Option zur Berücksichtigung der eingeschränkten Ausführungs- und Zugriffsrechte einiger Dateien stellt in den seltensten Fällen ein Risiko dar.
Wird durch Userprozesse in hohem Maß auf das */tmp(/users)*-Verzeichnis zugregriffen und pro Sitzung mehrere Megabyte dort abgelegt, sollte dieser Bereich vom NFS-Server hinzugemountet werden, um nicht durch überbeanspruchte Ramdisks den Betrieb zu gefährden. Hierbei ist jedoch zu beachten, dass zum einen Schreibzugriff auf diesen Bereich des Servers gewährt werden muss und dass zum anderen die gemeinsame Nutzung eines */tmp(/users)*-Verzeichnis sinnvoll gestaltet werden muss. Um zu verhindern, dass die Festplatte des Servers durch Überlastung des */tmp*-Bereichs überlaufen kann und den Betrieb stört, sollte dieser Bereich auf einer eigenen Festplatte oder Partition untergebracht sein.
Weiterhin können sich nicht mehrere Rechner das */tmp* ohne Anpassungen teilen, da dort viele Sockets installiert werden, die den Start oder Betrieb z.B. des X11, der Datenbanken und Applikationen verhindern könnten. Deshalb wurde ein Unterverzeichnis *users* geschaffen, in dem alle mengenlastigen Anwendungen, wie z.B. Netscape oder StarOffice ihren Cache schreiben. Weiterhin werden hier die Fehlerausgaben der X11-Sitzung eines jeden Users durch entsprechende Umleitung oder Konfiguration geschrieben. Einige Applikationen müssen also für diese spezielle Betriebsumgebung angepaßt werden.
Dieses Vorgehen hat zwei wichtige Vorteile: Zum einen sind alle Updates, die im Bereich der Applikationen erfolgen sofort und ohne weiteren Eingriff auch auf den DXS verfügbar, was den Administrationsaufwand für diese Bereiche gering hält. Zum anderen wirkt für beide Zugriffsarten auf diese Bereiche

[1] siehe *boot*-Skript im Anhang

des Serverfilsystems der Cachingmechanismus des Kernels. D.h. wird eine Applikation, ein Officepaket oder Netscape häufig aufgerufen, so beschleunigt sich der Start sowohl lokal als auch bei der Bereitstellung über das NFS und damit direkt für die betroffene DXS. Eine gute Speicherausstattung[2] des Servers trägt wie bereits angedeutet zu einer guten Performance eines DXS-LAN's bei.
Sollen jedoch DXS an einem Server einer anderen Software- oder Hardwarearchitektur betrieben werden, läßt sich das Sharing des Filesystems nur noch in Grenzen aufrecht erhalten. In einer solchen Situation bzw. angesichts des Preisverfalls für Speicherbausteine kann die Verwendung von dedizierten Bootservern für Thin-Clients sinnvoll werden. Wenn kein Benutzerbetrieb auf den Servermaschinen stattfindet, sinkt die Ausfallwahrscheinlichkeit dieser Maschinen und beeinflußt den Betrieb der DXS positiv, da Leistungsengpässe vermieden werden.

11.3 Mischbetrieb auf DXS

Diskless X-Stations können natürlich auch im Mischbetrieb als X-Terminals benutzt werden, wenn man dem grafischen Login einen Host-Chooser zur Auswahl der Maschinen mit grafischem Login vorschaltet. Dadurch kann der Administrator mehrere Ressourcen zur Auswahl stellen. So können seltener nachgefragte oder nicht im eigenen Netz betriebene Spezialserver unkompliziert zusätzlich zur Verfügung bereitgestellt werden. Vorstellbar wäre mittels SSH-Agent den Standarddesktop lokal auf der DXS zu starten und bestimmte aufwändige Appliklationen über das Netz von einem speziellen Server zu starten. So wird der Server von Standardaufgaben entlastet und der Speicher- und Ressourcenbedarf der DXS ufert nicht aus.

[2] 384 MByte-1 GByte, je nach Art des eingesetzten Softwarespektrums.

Kapitel 12

Die nächsten Schritte

Der bisherige Teil der Projekterläuterungen befaßte sich mit Software-Anpassungen und Hardwareimplementationen zur Einrichtung eines neuen Gerätetyps mit dem Ziel, den aufzubringenden Arbeitsaufwand gerade unter dem Gesichtspunkt steigender Maschinenzahlen in den Griff zu bekommen. In diesem Zusammenhang wurden notwendige Netzwerkprotokolle vorgestellt und die Verteilung der Aufgaben zwischen den Thin-Clients und ihren Servern diskutiert.
Wie gezeigt ist die zentrale Datei für die Konfiguration der Clients die */etc/dhcpd.conf*. Da in dieser Datei alle wichtigen Eigenschaften eines jeden Thin-Clients, ob X-Terminal oder Diskless X-Station abgelegt sind, lassen sich auf Basis dieser Datei weitere Automatisierungen vornehmen. Das lohnt sich insbesondere bei einer großen Anzahl von Geräten mit sehr verschiedenartigen Hardware- und Software-Konfigurationen. Weiterhin lassen sich auch die Erzeugung der DNS-Dateien, der Freigabelisten etc. auf diese Weise automatisieren.
Am besten setzt man hierzu eine Datenbank ein, bzw. erweitert die bereits vorhandene Hardwaredatenbank[1]. Möchte man eine zentrale Funktionsüberwachung der eingesetzten Geräte implementieren, kann auch hierfür die Datenbank Grundlage der Konfiguration sein.
Wenn bereits eine Konfiguration in der in den vorigen Kapiteln beschriebenen Weise besteht, können die bereits vorhandenen Informationen als Ausgangsbasis für die Datenbank dienen. Wie man die Daten der */etc/dhcpd.conf* in die Datenbank bringt, ist im Anhang beschrieben und anhand eines Beispielskriptes erläutert. Nun bietet jedoch die Einführung einer Datenbank eine ganze Reihe von Varianten, die über reine Konfigurationsaufgaben hinausgehen. Konzepte hierzu werden im nun folgenden Teil eingeführt.

[1] Dieses könnte zur Katalogisierung der eingesetzten Hardwarekomponenten dient.

Teil III

Datenbankgesteuerter Betrieb großer Rechnerpools

Kapitel 13

Einleitung

13.1 Aufbau des dritten Teils

Nach der Vorstellung von softwaretechnischen Lösungen zum festplattenlosen Betrieb einer größeren Zahl von Arbeitsplätzen soll nun eine Reihe weiterer Möglichkeiten erörtert werden, diesen Betrieb weitgehend automatisiert abzuwickeln. Zum zentralen Element wird eine SQL-Datenbank, die als Backend einer ganzen Reihe von Aufgaben dient.
Deshalb wird es in den nächsten Abschnitten darum gehen, die Auswahl eines geeigneten Datenbankproduktes zu motivieren (Kapitel 13), ihre nutzbaren Schnittstellen zu evaluieren, den Aufbau der Tabellen zu erläutern (Kapitel 14) und einige Werkzeuge zur Eingabe und Verwaltung der Daten vorzustellen (Kapitel 15). Zentrales Element ist die Einführung und sinnvolle Verknüpfung der benötigten Datenfelder. Hierzu wird eine Auswahl von Werkzeugen vorgestellt, die aus der Datenbank Konfigurationsdateien für unterschiedliche Bedürfnisse generieren kann, womit der Rückbezug auf den II. Teil des Projektes hergestellt wird. Die Zusammenfassung aller relevanten Betriebsdaten lädt nun dazu ein, weitere Verwendungen zu realisieren, wozu die betriebliche Erfassung und Inventarisierung zählen (Kapitel 18).
Ein weiteres Kapitel wird sich zum Ende dieses Teils mit den Fragen zur Systemüberwachung auseinandersetzen (Kapitel 17). Hier ergeben sich eine Reihe neuer Perspektiven, wenn es gelingt diesen zentralen Aspekt der Systemverwaltung mit den Fähigkeiten der Datenbank zu verknüpfen. Als Illustration wird eine kleine Auswahl an Software-Werkzeugen vorgestellt, für welche ebenfalls Beispielimplementationen zur Erstellung der Konfigurationsdateien existieren (Kapitel 16).

13.2 Vorteile des Datenbankeinsatzes

Datenbanken zur Erfassung und Bearbeitung großer Bestände an Informationen gehören zu den frühesten Anwendungsbereichen von elektronischen Rechnern. Die Verwaltung von vielen Einzeldaten, deren sinnvolle Verknüpfung, Auswertung und Weiterverarbeitung gehört zu den komplexen Aufgaben in betrieblichen Umfeldern. Hier gelingt es recht schnell, große Produktivitätsfortschritte zu erzielen.
Hohe Komplexität, die Verknüpfung und Verfügung über unterschiedliche Informationen zu den verschiedensten Fragestellungen zeichnen inzwischen auch Rechnernetze, Computerkomponenten und Software-Angebote einer Vielzahl von Arbeitsplätzen aus. Viele Informationen werden an mehreren Stellen benötigt und es existieren enge Abhängigkeiten. IP-Nummern, Domänen- und Hostnamen sind für DHCP, DNS und zur Inventarisierung bzw. Kennzeichnung wichtig. Eine bestimmte Hardwarekomponente, wie eine z.B. eine Grafikkarte unterstützt nicht nur einen gewissen Bustyp, in dem sie ausschliesslich funktioniert, sondern kann Monitore in einer bestimmten Auflösung und Farbtiefe ansteuern. Dazu benötigt die Software wiederum die Information, welcher Treiber in welcher Konfiguration geladen werden muss. Daneben interessieren natürlich auch Garantiezeiträume, Beschaffungsdatum und -preis, sowie die Information, wo entsprechende Bauteile im Einsatz ist.
Eine Datenbank bietet eine vernünftige Abstraktionsebene, um verschiedene Daten geeignet miteinander zu verknüpfen und für die unterschiedlichsten Anforderungen entsprechend verfügbar zu machen.
So ist man auf der einen Seite nicht auf ein einziges Frontend festgelegt, sondern kann dank des SQL-Standards über verschiedene Tools auf die Datenbank zugreifen. Export-Routinen erlauben es, die Daten schnell und einfach anderen Programmen und Software-Paketen zugänglich zu machen.
Für das gestellte Anwendungsgebiet kristallisieren sich drei Bereiche heraus.

Netzwerkadministration Hierzu zählt die Erstellung und Pflege der Konfigurationsdateien für die Thin-Clients und die automatische Installation von Linux-Servern und -Workstations. Mittels Datenbank lassen sich DNS-Tabellen verwalten und verschiedene kleine Hilfsroutinen, z.B. zur Erstellung der Exportlisten[1] des NFS-Servers durchführen. Aus Sicht der Netzwerkadministration läßt sich schließlich die Verwaltung von Schlüsseln zur Sicherung der Kommunikation (wie Host-Keys für die Secure Shell oder

[1] üblicherweise */etc/exports*

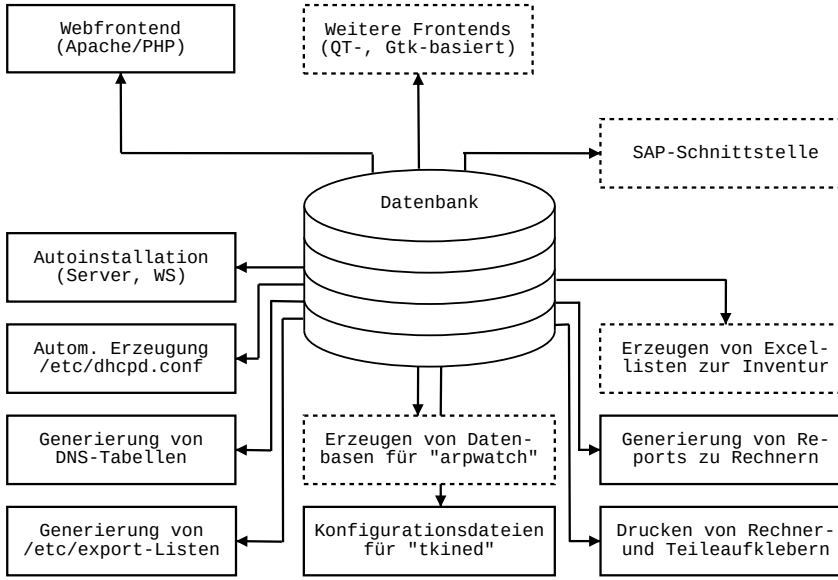


Abbildung 13.1: Einsatzmöglichkeiten der Datenbank

Schlüssel zur Absicherung des Datenverkehrs mittels IPSec[2]) realisieren.

Verwaltung Ein weiteres Anwendungsgebiet der Datenbank liegt im Bereich der Inventarisierung und Verwaltung der Rechnerbestände. Hier kommt es darauf an, leicht einen Überblick über Werte, Garantiezeiten, Einsatzorte etc. zu gewinnen. An dieser Stelle könnten Exportroutinen z.B. zur Erstellung von Exceltabellen zur Buchhaltung oder Schnittstellen zu SAP-Systemen interessant werden. Darüberhinaus kann die Kennzeichnung der eingesetzten Maschinen und ihrer Einzelteile mit der Datenbank verknüpft werden.

Monitoring Ein drittes Anwenungsfeld liegt in der System- und Netzwerküberwachung. Alle Systeme, die betriebsbereit sind, lassen sich der Datenbank entnehmen und so entsprechende Zuordnungen schaffen. Es ist denkbar, genaue Standortkoordinaten einzelner Rechner einzuführen, um diese in stark dezentralisierten Umgebungen einfach identifizieren zu können.

[2]Es gibt eine Reihe von Ansätzen den Datenverkehr auf IP-Ebene abzusichern. Hierbei implementiert IPSec die Datenverschlüsselung.

13.3 Wahl der Datenbank und ihrer Schnittstellen

Als Datenbank kommt MySQL[3] in Frage. Sie ist eine ständig weiterentwickelte Mid-Range-Database, die zwar nicht den vollen SQL-Umfang unterstützt, dafür aber völlig frei zur Verfügung steht und sehr gut dokumentiert ist. Es gibt gute Perl- und PHP-Schnittstellen.
Eine weitere freie Datenbank steht mit PostgreSQL[4] zur Verfügung. Auch hier stehen vielfältige Programmierschnittstellen zur Auswahl. PostgreSQL kennt den Datentyp IP-Nummer, was für die konkrete Anwendung von Vorteil sein kann.
MySQL erhielt wegen der größeren Verbreitung und den einfacher zu handhabenden Steuertools den Vorzug.

13.4 Generierung von Konfigurationsdateien

Mit der Einführung einer Datenbank als zentrales Konfigurationstool wird eine weitere Abstraktionsebene eingeführt. Damit erhöht sich der Komplexitätsgrad bei der Einrichtung der Diskless X-Stations und X-Terminals.
Der Vorteil der höheren Abstraktionsebene liegt in der verbesserten Verfügbarkeit der Daten. Diese liegen nun nicht mehr in einer komplexen Konfigurationsdatei vor, sondern stehen in verallgemeinerter Form zur Verfügung. Damit beschränken sich die Gestaltungsmöglichkeiten nicht nur auf Thin-Clients, die mittels DHCP betrieben werden, sondern erlauben auch ein einfaches Aufsetzen von Servern und Workstations. Quasi nebenbei können nun Aufgaben zur Erstellung von DNS-Tabellen und Systemmonitoring miterledigt werden.

13.5 Aufbau der Datenbank

Die Datenbank ist in ihren wichtigen Bereichen normalisiert, um Redundanzen zu vermeiden und den Tabellenplatz effektiv zu nutzen. Randaspekte, welche in der hier vorgestellten Lösung eine untergeordnete Rolle spielten, werden direkt aufgenommen und nicht über weitere Tabellen verknüpft.

[3] Siehe hierzu [W8]. Auf dieser Website finden sich eine umfassende Dokumentation und weiterführende Links zum Thema, wie Mailinglisten, FAQ's ... [M4] und [M9] bieten einen guten Einstieg in die Programmierung relationaler Datenbanken am Beispiel von MySQL.

[4] Siehe hierzu [W9]. Von dieser Website aus gibt es die Verzweigungen zur Suchfunktion, Dokumentation ...

Einige Datensätze, wie Informationen zu Eigentümern, den Herstellern von Hardware, Lieferanten und Zuständigen für Organisationen, Abteilungen, Rechnergruppen und einzelne Server werden gemeinsam in einer Tabelle zusammengefaßt, da sie eine sehr ähnliche Struktur aufweisen.

Kapitel 14

Die Struktur der Datenbanktabellen

14.1 Überblick

Die Konfigurations-, Verwaltungs- und Inventardatenbank (Hardwareverwaltung) besteht aus zwei Haupttabellen: Der Liste aller Rechner (Tabelle Rechner) und die Liste aller Hardwarekomponenten, die in Rechnern verwendet oder im Netz eingesetzt werden können (Tabelle Hardware). Diese Grundtabellen sind um weitere ergänzt worden. Diese beschreiben einzelne Hardwarekomponenten näher und gruppieren diese ihrem jeweiligen Typ zu bzw. ordnen Rechner in definierte Gruppen. Optionale Eigenschaften werden in den jeweiligen Property-Tabellen hinterlegt. Die meisten Tabellen enthalten sehr umfangreiche Informationen zu einzelnen Objekten, auf die eine Reihe Werkzeuge mit sehr verschiedenen Aufgabenstellungen den Zugriff erhält.

14.2 Tabelle "HardWare"

Eines der Kernstücke in der Datenbank ist die Tabelle zur Verwaltung der verwendeten Hardware. Diese enthält alle "unteilbaren" Komponenten, die im Rechnernetzwerk eingesetzt werden können und in einem bestimmten Umfang miteinander kombinierbar sind. Das umfaßt z.B. denkbare Kombinationen aus Mainboard, CPU und Speicher, dem Anschluß bestimmter Peripheriegeräte etc. Wenn auch der Trend zu Beschaffung von Rechnern großer Anbieter mit längeren Garantiefristen und höher integriereten Systemen geht, so werden diese Maschinen ebenfalls erweiterbar sein, bzw.

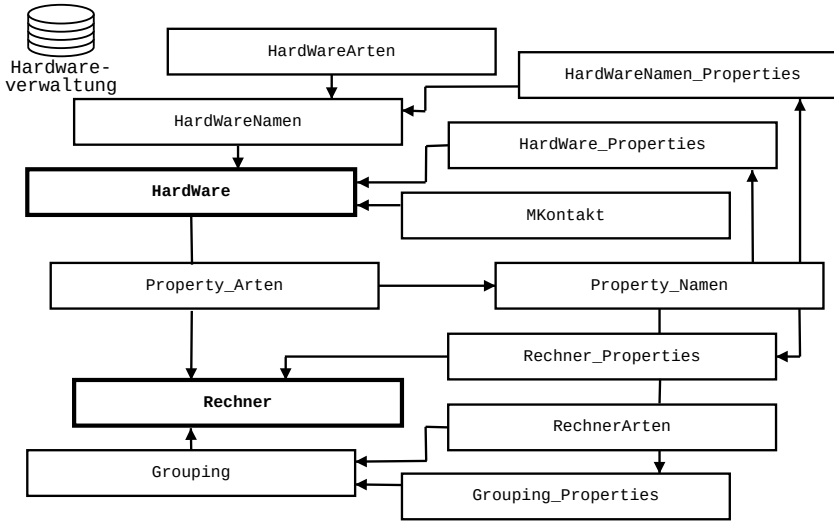


Abbildung 14.1: Zuordnung der Datenbanktabellen

wird auch nach diesem Zeitraum der Austausch einzelner Komponenten zur Erhaltung der Betriebsfähigkeit notwendig werden. Komplettsysteme nicht standardisierter Hardware werden entsprechend ihrer Hauptkomponente erfaßt, was im Fall der einfachen Office-Geräte der großen Hersteller das hochintegrierte Mainboard mit Netzwerkadapter, Grafik- und Audiounterstützung onboard sein wird. Diese Konstellationen sind beim Design der Konfigurationsskripten zu berücksichtigen, stellen jedoch kein Problem bei der nahtlosen Integration in das vorgelegte Konzept dar.

14.2.1 Die Haupttabelle

Der Aufbau der Tabelle sieht wie folgt aus:
Der Tabelle "HardWare" sind die Tabellen "HardWareNamen", die die Bezeichnungen einzelner Hardwarekomponenten enthält, "LieferantenID" und "HardWare_Properties" zugeordnet.
Die Tabellenfelder "Beschafft", "Garantie_bis", "BruttoPreis" und "StrichCode" sind für die Inventarisierung notwendige Felder. Das erste Feld dieser Aufzählung nennt den Beschaffungstag, das nächste bezeichnet den Zeitpunkt des Ablaufes der Garantiefrist. Es folgt der Bruttopreis zum Zeitpunkt der Beschaffung. Der "StrichCode" bezeichnet eine eindeutige Identifikationsnummer für ein bestimmtes Teil, die als Aufkleber mit Strich- und Zahlenkennung auf das Hardwareteil aufgebracht wird. Ausführungen

Tabellenfeld	Typ des Feldes	Beschreibung
HardWareID	Integer	Identitätsnummer eines Teils
HardWareNameID	Integer	ID-Feld zur Namentabelle
RechnerID	Integer	ID-Feld zur Rechnertabelle
LieferantID	Integer	ID-Feld zur Lieferantentabelle
Garantie_bis	Datum	Ablaufdatum der Garantiefrist
StrichCode	String (255 Zeichen max.)	Interner eindeutiger Bezeichner
BruttoPreis	Fließkommazahl	Anschaffungspreis
Beschafft	Datum	Beschaffungstag
EigentuemerID	String (255 Zeichen max.)	Besitzeridentifikationsnummer des Gerätes

Tabelle 14.1: HardWare

hierzu findet man im Abschnitt zu den Inventarisierungswerkzeugen.

14.2.2 Tabellen "HardWareNamen"

Jedes verbaute Hardwareteil verfügt über bestimmte Eigenschaften, die aus seinen Leistungsdaten und den verfügbaren Schnittstellen bestehen. Diese Informationen bestimmen die Einträge für die verschiedenen Konfigurationsdateien.

Tabellenfeld	Typ des Feldes	Beschreibung
HardWareNameID	Integer	Identifizierungsnummer
Name	String (255 Zeichen max.)	Bezeichnung der Hardware
HardWareArtID	Integer	Art der HardWare (Monitor, Maus, ...)
Hersteller	String (255 Zeichen max.)	Name des Herstellers

Tabelle 14.2: HardWareNamen

Hierzu gehören bei Monitoren die maximale Bildschirmauflösung, die Vertikal- und Horizontalfrequenz. Jede Netzwerkkarte bringt als Eigenschaft ihre MAC[1]-Adresse mit, die die Grundlage des hier verwendeten DHCP[2] bil-

[1] Media Access Control
[2] Dynamic Host Control Protocol

det.
Fast jedes Hardwareteil erfordert bestimmte Treiber, damit es mit der Software zusammenarbeiten kann. Für die Grafikkarte muss der entsprechende XFree86-Treiber ausgewählt und nach den Parametern von Monitor und Grafikkarte konfiguriert werden. Dazu muss weiterhin die Art der Maus bekannt sein - z.B. ob es sich um eine PS/2, eine serielle Maus oder eine mit Scroll-Rad handelt. Bei der Tastatur sind Informationen zur Sprache entscheidend.

14.2.3 Tabelle "HardWareArten"

Für den einfacheren Überblick erfolgt eine Kategorisierung der einzelnen Hardwareteile nach Arten. So wird nach Mäusen, Tastaturen, Gehäusen, Monitoren, Adapterkarten etc. unterschieden. Die Teile dieser Kategorien sind üblicherweise einzeln erhältlich und unterscheiden sich in Funktion und Eigenschaften von den jeweils anderen.

Tabellenfeld	Typ des Feldes	Beschreibung
HardWareArtID	Integer	Identifizierungsnummer
Art	String (255 Zeichen max.)	Bezeichnung der Hardware

Tabelle 14.3: HardWareArten

14.2.4 Eigenschaftentabelle "HardWare_Properties"

In der Tabelle "HardWare_Properties" werden Zusatzinformationen, die für ein bestimmtes Hardwareteil spezifisch sind, wie z.B. die MAC-Adresse einer Netzwerkkarte, oder der herstellerseitige Strichcode als Zusatzinformation zur Identifizierung gespeichert. Das Aussehen dieser Tabelle kann man dem Abschnitt "Property-Tabellen" entnehmen.

14.2.5 Tabelle "MKontakt"

Diese Tabelle umfaßt alle auftretenden Kontakte. Hierzu zählen Eigentümer der aufgestellten Maschinen, Beauftragte für Rechnergruppen, Verantwortliche für bestimmte Server, Hersteller und Lieferanten. Die Unterscheidung geschieht mittels der zugeordneten Tabelle "KontaktArten", welcher ihrerseits die beiden Spalten für die obligatorische Identifikationsnummer und die Bezeichnung für den Kontakt enthält.

Tabellenfeld	Typ des Feldes	Beschreibung
MKontaktID	Integer	Identifikationsnummer
Name	String (100 Zeichen max.)	Kurzbezeichnung des Kontaks
Name_Lang	String (255 Zeichen max.)	Ausführliche Bezeichnung (Titel, Firmenname etc.)
KontaktArtID	Integer (klein)	Art des Kontaktes

Tabelle 14.4: MKontakt

14.3 Tabelle "Rechner"

14.3.1 Die Haupttabelle

Die Zentraltabelle der Datenbank bildet "Rechner". Hier sind alle im Netzwerk zu managenden Geräte eingetragen: Server, Diskless X-Stations, X-Terminals und administrierbare Switches.

Tabellenfeld	Typ des Feldes	Beschreibung
RechnerID	Integer	Identitätsnummer eines Rechners
IPAddress	String (15 Zeichen)	IP-Adresse des Rechners
HostName	String (255 Zeichen max.)	Hostname des Rechners
GroupingID	Integer	Zuordnung zu einer Gruppierung

Tabelle 14.5: Rechner

Die Rechner werden wiederum Gruppen zugeordnet (Tabelle "Grouping") und verfügen über zusätz-liche Eigenschaften, die über die Stammdaten wie IP-Adresse und Hostname hinausgehen (Tabelle "Rechner_Properties" oder "Grouping_Properties").

14.3.2 Tabelle "Grouping"

Die Tabelle "Grouping" bestimmt mit der Haupttabelle alle Netzwerkeigenschaften der zu managenden Knoten. Sie verfügt dazu über eine Reihe von Basisdaten:
Weitere Eigenschaften können über die Zusatztabelle "Grouping_Properties" angefügt werden.

Tabellenfeld	Typ des Feldes	Beschreibung
GroupingID	Integer	Identitätsnummer der Gruppierung
Name	String (100 Zeichen max.)	Name der Gruppierung
NetMask	String (15 Zeichen)	Netzmaske (IP)
DomainName	String (100 Zeichen max.)	Name der Domain
GateWay	String (15 Zeichen)	Gateway-Adresse (IP)
FirstDNS	String (15 Zeichen)	Erster DNS-Server (IP)
SecondDNS	String (15 Zeichen)	Zweiter DNS-Server (IP)
Broadcast	String (15 Zeichen)	Broadcast-Adresse (IP)
RechnerArtID	Integer	Art des Rechners (DXS, Server...)

Tabelle 14.6: Grouping

14.3.3 Tabelle "Rechner_Properties"

Eigenschaften die nur ein ganz bestimmtes System im Netzwerk betreffen, können hier hinterlegt werden. Diese können entweder ergänzenden oder überschreibenden Charakter gegenüber den Gruppeneigenschaften haben. Den Aufbau dieser Tabelle entnehme man dem Abschnitt "Property-Tabellen".

14.3.4 Die Property-Tabellen

Die Property-Tabellen haben die Aufgabe, zusätzliche aber nicht zwangsläufig vorhandene Eigenschaften von Hardwareteilen, Rechnern bzw. Gruppen aufzunehmen. Der Bezug zu den Einträgen in den anderen Tabellen wird über das zweite Feld hergestellt. Die Bezeichnung der ersten beiden Tabellenfelder ergeben sich aus den Namen der zugehörigen Haupttabellen.
In einigen Bereichen kann es zu Überschneidungen von Eigenschaften kommen: So könnten bestimmte Werte für eine Gruppe von Rechnern gemeinsam definiert werden, aber für einen einzelnen Rechner aus dieser Gruppe überschrieben werden, weil er für wenige Variablen eine Sonderstellung einnimmt. Dasselbe mag für der Hardware zugeordnete Eigenschaften gelten.

Tabellenfeld	Typ des Feldes	Beschreibung
(Tabellenname) PropertyID	Integer	Identitätsnummer des Eintrags
(Tabellenname)ID	Integer	Zuordnung zum entsprechenden Item
PropertyNameID	Integer	Name der Eigenschaft
Value	String (255 Zeichen max.)	Eintragung des entsprenden Eigenschaftswertes

Tabelle 14.7: Eigenschaften

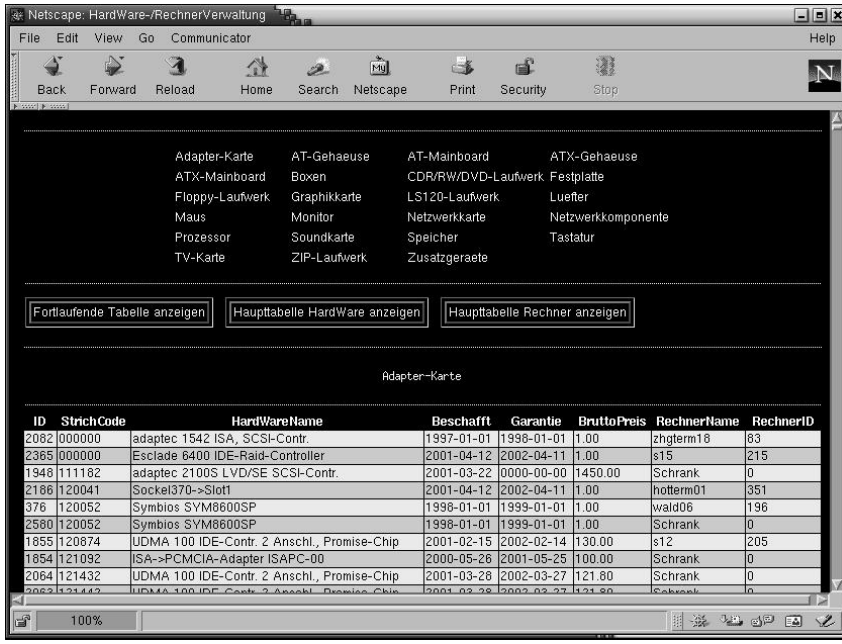
Ein Monitor hat überlicherweise eine optimale Auflösung, welche jedoch durch die Eigenschaften der Grafikkarte nicht erreicht oder übersteuert werden könnten oder weil einige Maschinen für Personen mit eingeschränktem Sehvermögen reserviert werden usw. Diese Differenzierung kann ohne Probleme in der Datenbank hinterlegt werden, jedoch liegt die Aufgabe der korrekten Umsetzung in der Gestaltung des Konfigurationsskriptes, da dieses sich schlecht auf die Datenbank abbilden läßt

Kapitel 15

Benutzerschnittstellen

15.1 Varianten der Datenein- und Ausgabe

In diesem Abschnitt werden einige Möglichkeiten für Benutzerschnittstellen zur gewählten Datenbank vorgestellt. Exemplarisch wird auf die PHP-basierte Webschnittstelle etwas näher eingegangen. Jedoch handelt es sich bei der vorliegenden Implementierung um eine spezifische Anpassung an die Bedingungen zur Verwaltung der Hardware- und Rechnerbestände des Studierendennetzes an der Universität Göttingen.
Viele kommerzielle Datenbanken enthalten in ihrem Lieferumfang eine ganze Reihe von Werkzeugen zur Erstellung von Eingabemasken oder grafischer Frontends. Dies gilt jedoch nur eingeschränkt für die frei verfügbaren Datenbanken.
Für den Zugriff auf die Informationen in der Datenbank kommen deshalb verschiede Werkzeuge in Frage, die auf die Datenbank über deren externe Schnittstellen zugreifen. Eine ganze Reihe der hier vorgestellten Werkzeuge, wie sie im Zuge des Projektes erstellt wurden, laufen automatisiert bzw. teilautomatisiert auf den Servern ab. Dazu zählt die Generierung der Konfigurationsdateien des DHCP, DNS und der diversen Überwachungstools. Dies geschieht meistens ohne Benutzerinteraktion, so dass sich Kommandozeilentools besonders anbieten. An anderen Stellen jedoch sind es nicht die Techniker und Systemadministratoren, welche mithilfe der Datenbank ihre Aufgaben erfüllen. Buchhaltungen und Verwaltungen arbeiten häufig mit anderen Betriebssystemen und Anwendungsprogrammen, u.a. mit "Access"



ID	StrichCode	HardWareName	Beschafft	Garantie	BruttoPreis	RechnerName	RechnerID
2082	000000	adaptec 1542 ISA, SCSI-Contr.	1997-01-01	1998-01-01	1.00	zhgterm18	83
2365	000000	Esclade 6400 IDE-Raid-Controller	2001-04-12	2002-04-11	1.00	s15	215
1948	111182	adaptec 2100S LVD/SE SCSI-Contr.	2001-03-22	0000-00-00	1450.00	Schrank	0
2186	120041	Sockel370->Slot1	2001-04-12	2002-04-11	1.00	hotterm01	351
376	120052	Symbios SYM8600SP	1998-01-01	1999-01-01	1.00	wald06	196
2580	120052	Symbios SYM8600SP	1998-01-01	1999-01-01	1.00	Schrank	0
1855	120874	UDMA 100 IDE-Contr. 2 Anschl., Promise-Chip	2001-02-15	2002-02-14	130.00	s12	205
1854	121092	ISA->PCMCIA-Adapter ISAPC-00	2000-05-26	2001-05-25	100.00	Schrank	0
2064	121432	UDMA 100 IDE-Contr. 2 Anschl., Promise-Chip	2001-03-28	2002-03-27	121.80	Schrank	0

Abbildung 15.1: Darstellung der Hardwaredaten

und "Excel"[1]. Hier bietet sich die Benutzung der ODBC-Schnittstelle[2] an, welche sowohl die hier verwendete populäre MySQL-Datenbank, als auch PostgreSQL anbieten.
Mittels dieser Schnittstelle gelingt die Verknüpfung einer ganzen Reihe weiterer Applikationen auf die Bestände der Datenbank. ODBC verknüpft die leichte Handhabbarkeit der Datenbank unter einem Open-Source-Betriebssystem mit der Verwendbarkeit von Standardtools außerhalb der Welt der Systemadministration. So können Programmiererfahrungen z.B. mit "Access" und "Visual-Basic" dazu benutzt werden, einfache Interfaces als Tabellen oder Formulare zu generieren, die eine Einbindung der Daten in Berichte oder Kalkulationen erlauben. Zusatzkosten für Software fallen nicht oder in sehr geringem Maße an, Mitarbeiter müssen nicht aufwändig umgeschult werden. Die Installation des ODBC-Treibers bereitet keine Schwierigkeiten,

[1] MS-Access ist ein Datenbankprogramm und MS-Excel eine Tabellenkalkulationssoftware aus dem Hause Microsoft, welches überlicherweise in einem Officepaket ausgeliefert wird.

[2] Das Open Database Connect Protocol bietet eine von Microsoft definierte Programmier-API (Application Interface) zum Zugriff auf Datenbanken.

anschließend sind die Verbindungsparameter zum Zugriff auf die konkrete Hardwaredatenbank vorzunehmen und einige Einstellungen evtl. anzupassen.
Eine weiterer Weg der Dateneingabe besteht in der Auswertung bereits installierter DHCP-Konfigu-rationen. Im hier beschriebenen Projekt erfolgte die Verknüpfung mit einer Datenbank erst nachdem bereits eine gewisse Anzahl an Diskless Clients betrieben wurde. Die Einstellungen in der *dhcpd.conf* wurden zur Bestückung der Datenbank herangezogen, da viele Informationen durch die Auswerung der MAC-Adresse bereits einzelnen Maschinen zugeordnet werden konnten. So entfiel eine aufwändige händische Eingabe und die Migation auf die Datenbanklösung wurde erleichtert. Das Beispielskript hierzu, in Perl[3] geschrieben, findet sich im Anhang (Abschnitt D.5).
Eine gute plattformübergreifende Lösung gelingt durch die Erstellung eines Webfrontends. Die notwendigen Werkzeuge, der Webserver Apache mit einem Plugin zur Auswertung von PHP-Skripten, sind frei verfügbar, gut dokumentiert und bei jeder Linuxdistribution dabei. Mittels solcher Schnittstellen läßt sich eine gute Anpassung an die jeweiligen Bedürfnisse erreichen, jedoch steigt der Programmieraufwand, wenn nicht sehr generische Interfaces, die zu Lasten der Ergonomie gehen, verwendet werden.

15.2 Die PHP/Webschnittstelle

Die PHP-Skriptsprache enthält die notwendigen Datenbankschnittstellen und eignet sich zur Programmierung komfortabler Webschnittstellen. Die Erstellung der HTML-Dokumente erfolgt auf Serverseite, so dass keine hohen Anforderungen an die Webbrowser gestellt werden, die als Display-Werkzeug zum Einsatz kommen.
Die Seitenbeschreibungssprache HTML bietet eine gut strukturierbare Tabellendarstellung, jedoch sind die Beschränkungen durch die darstellbare Breite im Browserfenster zu beachten. Ein weiteres Problem liegt in der Länge der angezeigten Listen. So kommt es nicht nur durch notwendiges Blättern zum Ergonomieverlust für den Benutzer, sondern es bereiten die zunehmenden Dokumentenlängen den Browsern Schwierigkeiten beim Rendering. Um dieser Schwäche zu begegnen, wurde die Möglichkeit geschaffen, auf einzelne Gruppen innerhalb der Liste zugreifen zu können. Auf diese Weise können über den Tabellenkopf direkt Gruppen von Rechnern bzw. Hardwarearten angesprungen werden. Ein weiterer Klick erlaubt dann

[3] wie schon eingangs der Arbeit angemerkt, eignet sich die Skriptsprache Perl hervorragend zur Bearbeitung von Textstrings und zum Zugriff auf Datenbanken. Die Grundlagen zur Perl-Programmierung sind [M8] entnommen.

Abbildung 15.2: Darstellung der Rechnerliste

die ausschließliche Darstellung dieser Teilliste, wobei dann der Tabellenkopf zum Springen zwischen den Teillisten umkonfiguriert ist.
Einige Standardfunktionen, wie das Sortieren nach einzelnen Tabellenspalten lassen sich unproblematisch umsetzen. Die Tabellenköpfe können als klickbare "Reiter" gestaltet werden. Schwieriger wird es hingegen aufwändige Such- und Filterfunktionen zu implementieren und die unterschiedlichen Eingabemasken zu gestalten. Hierin liegt ein Nachteil dieses Ansatzes, da generalisierte Schnittstellen zwar zügig zu erstellen sind, jedoch selten bedarfsgerecht ausfallen. Die Orientierung an Ergonomiegesichtspunkten erfordert Detailarbeit, die bei notwendigen Veränderungen erneut zu leisten ist. Beschränkungen liegen weiterhin in den Realisierungen des HTML-Formulares,

da jeder Button, der dargestellt werden soll, wie im vorgestellten Beispiel zum Umschalten zwischen Rechner- und Hardwarelisten und ihrer Darstellung in geschlossener Langform, mittels solcher Formularabfragen gestaltet werden muss.

Kapitel 16

Datenbankgesteuerte Administration

16.1 Erweiterte Möglichkeiten

Sinnvollerweise wird man in der Datenbank nicht nur die Informationen zu den Thin-Clients ablegen, sondern auch die im Einsatz befindlichen Server und Workstations inventarisieren. Das erleichtert nicht nur die Bestandsaufnahme aller eingesetzten Rechner, sondern erlaubt auch die zentrale Erzeugung der Einträge für das Domain Name System (DNS).
In den nachfolgenden Abschnitten sollen einige der erweiterten Administrationsmöglichkeiten vorgestellt werden:

- Anpassung der Betriebsumgebung der Thin-Clients (Abschnitt 16.2)
- Automatisierte Installationsroutinen für Server und Workstations (Abschnitt 16.3), incl. der Entwicklung einer Wartungs- und Serviceplattform
- Konfiguration des Netzwerkes für das Domain Name System (Abschnitt 16.4)

Die hier vorgestellten Tools wurden für den konkreten Einsatz an der Universität Göttingen geschrieben und sind dementsprechend stark auf die Gegebenheiten zugeschnitten. Sie stellen daher Perspektiven und Varianten vor, wie ausgehend von der geschaffenen Datenbankplattform aus weitere Administrationsaufgaben realisiert werden können.
Genauso wäre es denkbar, ein Netz von klassischen Workstations mittels Datenbank direkt über die Installation und die späteren Updates zu steuern,

wie es im Abschnitt zur automatischen Serverinstallation beschrieben wird. Auch kann man sich eine Hybridlösung aus Festinstallation und Einrichtung per DHCP vorstellen, die Elemente der Konfiguration der Thin-Clients mit der Serverinstallation verknüpft. Das zentrale Anliegen der Vereinfachung der Steuerung größerer Rechnerpools läßt sich nun in vielfältigerer Weise erreichen.

16.2 Konfiguration der Thin-Clients

Durch die Bündelung aller zentralen Konfigurationsaspekte des Betriebes der Thin-Clients in der */etc/dhcpd.conf* genügt es, diese Datei mit den Informationen aus der Datenbank zu bestücken. Ein ähnliches Konzept wird in [Z7] beschrieben. Hierbei läßt sich diese Datei durch die Gruppierungen in der Datenbank für einen besseren Überblick strukturieren. Es wird jedoch immer noch eine Reihe von Variablen übrig bleiben, die nicht oder nur mit hohem Aufwand aus der Datenbank bestückt werden können. Hierzu könnten einige PnP-Konfigurationsdateien zählen, die in Abhängigkeit der eingesetzten Hardware erstellt werden und unabhängig auf jeder Maschine vorliegen.
Die Abbildung zeigt, wie die Konfiguration der Thin-Clients nun erfolgt. Damit entfällt die manuelle Abstimmung der Konfigurationsdateien zwischen verschiedenen DHCP-Servern und die Fehlerwahrscheinlichkeit unterschiedlicher Konfigurationen sinkt. Kommt es nun zu ungültigen Einstellungen, sind nicht mehr die Konfigurationsdatei des DHCP-Servers (*/etc/dhcpd.conf*) und davon abhängige Dateien zu konsultieren, sondern die Datenbank auf ihre Richtigkeit zu überprüfen.
Die Generierung der */etc/dhcpd.conf* geschieht am **cron**[1]-gesteuert mittels des Skriptes **dhcp-cron**, welches im Anhang (Abschnitt D.4) angegeben ist. Das Skript sorgt gleichzeitig für eine Überprüfung der Korrektheit der erzeugten Datei und die Funkionsfähigkeit der Serverumgebung. Damit wird das Konzept der redundanten Bootserver beibehalten. So läßt sich ein "single point of failure" vermeiden, der durch den Ausfall der Datenbank eintreten würde. Das Broadcastmodell des DHCP eignet sich besser und arbeitet effizienter als die Kaskadierung mehrerer Abfragen an verschiedene Datenbanken, falls eine nicht erreichbar sein sollte.

[1] Der Cron-Daemon erledigt auf unixbasierten Systemen den Aufruf wiederkehrender Aufgaben zu bestimmten Uhrzeiten, Tagen, Wochen und Monaten.

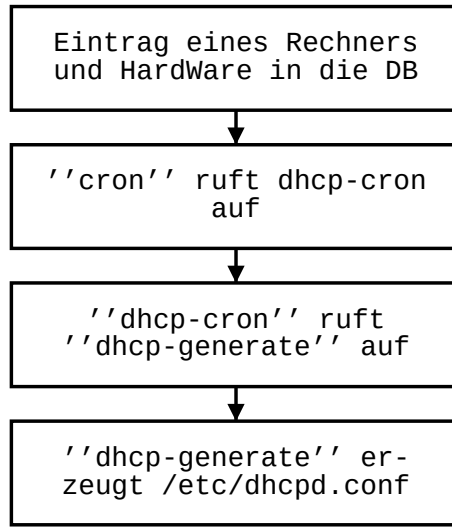


Abbildung 16.1: Ablauf der Konfiguration

16.3 Automatische Installation

16.3.1 Überblick

Mit den meisten Linuxdistributionen stehen inzwischen mehr oder weniger leistungsfähige und automatisierte Installationsroutinen zur Verfügung. Daher gibt es bereits einige Konzepte, um die Erstinstallation von Linux-PC zu Vereinfachen: Eines dieser Tools, Alice[2], wurde von SuSE entwickelt. Es benutzt zur Installation Listen von RPMs[3], welche auf dem Zielsystem eingespielt werden und eine Skriptliste, die systemspezifische Anpassungen übernimmt. Dieses Konzept eignet sich zur Neuinstallation, wenn nicht eine ganze Reihe spezieller Anpassungen an die Zielumgebung notwendig sind und alle einzusetzende Software in Form der Redhat-Package-Manager-Pakete vorliegt. Sind jedoch eine große Zahl von Veränderungen an den Standardpaketen erfolgt, zusätzliche Software installiert und spezielle Pfade und Verzeichnisse, z.B. für den Betrieb der DXS, notwendig, erzielt eine andere Herangehensweise bessere Erfolge. Diese soll im folgenden kurz besprochen werden.
Die Software der Server wird durch eine automatische Installation aufge-

[2] siehe Linux Magazin S.97 9/2001
[3] Software-Pakete im Redhat-Package-Manager-Format

spielt und durch automatische Updates auf dem aktuellen Stand gehalten.

16.3.2 Server-(Erst-)Installation

Beim Design einer Arbeitsumgebung zur Servererstinstalltion gelten andere Voraussetzungen als beim Massenbetrieb von Thin-Clients. Da Neuinstallationen von Rechnern im Verhältnis zum Bootvorgang der festplattenlosen Maschinen relativ selten auftreten, kann auf eine starke Redundanz verzichtet werden, weshalb es akzeptabel erscheint, die Daten direkt aus der Datenbank zu beziehen. Leistungsschwankungen und Verzögerungen bei Antwortzeiten sind aufgrund der Laufzeit des Installationsvorganges akzeptabel.
Mischlösungen aus einer automatisierten Installation des Dateisystems und einer dynamischen Konfiguration mittels DHCP sind darüberhinaus vorstellbar. Dabei sollte jedoch beachtet werden, dass wichtige Server im Netzwerk nicht von anderen (in diesem Fall dem DHCP-Server) abhängig sein sollten und besser über eine statische Konfiguration verfügen. Für bestimmte Workstations jedoch, die mit einer Festplatte ausgerüstet sind, kann eine solche Hybridlösung sinnvoll sein.

16.3.3 Wartungsumgebung auf Basis der DXS

Mit der Infrastruktur für die Diskless X-Stations besteht die Möglichkeit, neben der im letzten Kapitel beschriebenen Erstinstalltion auch eine bequeme Serviceumgebung zu betreiben. Unter der Voraussetzung, dass die Systemhardware funktioniert, ein Diskettenlaufwerk (oder bei Bedarf ein bootfähiges CD-Laufwerk) verfügbar ist, kann jede PC-Hardware aus dem Netz gebootet werden.
Eine DHCP-Umgebung ist hierfür nicht erforderlich. Damit besteht der Vorteil nicht auf ein gemeinsames physikalisches Netz mit dem Server angewiesen zu sein. Der Kernel, welcher zum Booten der DXS Verwendung findet, wird vor dem Tagging auf eine Free-DOS-Diskette geschrieben. Dieser Kernel wird dann mittels **loadlin** gestartet und erhält seine IP-Konfiguration und Startparameter aus einer Konfigurationsdatei, welche auch auf der Diskette liegt. Wenn die NFS-Freigaben entsprechend erfolgen, besteht von jeder Stelle des Netzwerkes aus eine bequeme Wartungsumgebung, welche alle notwendigen Tools enthält. Auf diese Weise lassen sich die Dateisysteme von Servern überprüfen, Backups anlegen und allgemeine Hardwarechecks durchführen, ohne dass das Betriebssystem von der Festplatte gestartet werden muss. Wurde das System erfolgreich aus dem Netz gebootet, kann der Zugriff für alle weiteren Handlungen über das Netzwerk erfolgen. So muss nicht für jeden Wartungsgang ein Administrator vor Ort sein, sondern läßt sich auch weniger erfahrenes bzw. geschultes Personal, wie z.B. der Benutzer

des entsprechenden Rechners, einsetzen. Linux erlaubt dem Administrator sich remote einen umfassenden Überblick über die eingesetzte Hardware zu verschaffen. Die Startdiskette könnte auch mittels eines einfachen Benutzerinterfaces erstellbar sein, so dass für den Administrationsfall eine Notstartdiskette bereits vor Ort ist.
Der Vorteil dieser Lösung liegt in der Aktualität und bequemen Ergänzbarkeit des Systems, wie sie für den Betrieb der Diskless X-Stations erforderlich ist. Dieses kann mittels speziell angepaßter CD's (hier ist der Umfang des Filesystems beschränkt) oder Disketten bei weitem nicht so komfortabel gestaltet werden. Hinzu kommt die Problematik permanenter Updates, welche zeitgleich zum Update der Server für diese Spezialmedien erfolgen müßten. Der klassische Nachteil liegt, wie bereits an anderer Stelle geschildert, in der Abhängigkeit von einer funktionierenden Netzwerkinfrastruktur (die jedoch die Grundvoraussetzung der hier geschilderten Betriebsumgebung bildet).

16.3.4 Automatische Updates

Ein klassisches Problem festplattengestützter Workstations liegt in der Divergenz ihrer Filesysteme im Laufe der Betriebszeit. Im Gegensatz zu den Diskless X-Stations wird man andere Wege beschreiten müssen, um die Software zwischen verschiedenen Maschinen konsistent zu halten. Dies geschieht am einfachsten, indem alle entscheidenden Teile des Dateisystems der betroffenen Rechner gegen das Filesystem eines Masterrechners abgeglichen werden. Seit einiger Zeit steht hierfür im Bereich der Open-Source-Software das Tool **rsync** zur Verfügung, welches über umfangreiche Mechanismen zur Steuerung und Reduzierung der Datenlast bei der Netzwerkübertragung verfügt. Dazu wurde ein eigener TCP-Dienst - RSYNC - definiert.
Hierbei muss jedoch beachtet werden, dass bestimmte Teile des jeweiligen Systems von diesem Update ausgenommen und anders behandelt werden müssen. Das betrifft zum einen alle hostspezifischen Einstellungen[4] und den Bereich der variablen Runtime- und Log-Dateien.
Ein Beispielskript zur Abhandlung solcher Updates wird im Anhang (Abschnitt F.2.3) beschrieben. Es basiert auf der Skriptsprache Perl und verwendet den Netzwerkdienst RSYNC zum Überprüfen und Kopieren der einzelnen Dateien. Mit RSYNC wurde ein TCP-basiertes Tool zum effizienten Abgleich von Dateisystemen geschaffen, das sich auf den Transfer von Unterschieden beschränkt und dadurch Netzlast und Dauer des Vorganges erheblich absenkt. Verschiedene Sicherheitsmechanismen, wie SSH-Verschlüsselung oder Passwortauthentifizierung sind implementiert. RSYNC ist in der Lage, mit allen Dateitypen umzugehen und die entsprechenden Zu-

[4] IP-Konfiguration, Rechnername, Dienstekonfiguration ...

griffsrechte beizubehalten. Damit gleicht es zum einen den Nachteil von **scp** bzw. **rcp** aus, welche nicht für solche Aufgaben optimiert sind und umgeht zum anderen Probleme, die beim Einmounten des Serverfilesystems und dem anschließenden Kopieren entstehen.

16.4 Erzeugung von DNS-Tabellen

Ein zentrales Element in großen TCP/IP-basierten Rechnernetzwerken ist das Domain Name System. Es erlaubt nicht nur dem Administrator einen leichteren Überblick, sondern hilft auch den Benutzern sich leichter zu orientieren. In vielen lokalen Netzen hat es sich als Standard durchgesetzt, einen DNS einzurichten.
Die Pflege der Datenbank bestimmt über die Güte des Ganzen. Durch die Automatisierung lassen sich die DNS-Tabellen stets aktuell halten. Gleichzeitig sinkt die Gefahr, bei den Einträgen Fehler zu machen, da dies nun von Skripten übernommen wird.
dns-generate erzeugt auf einem gegebenen DNS-Server alle notwendigen DNS-Dateien. Über die Eigenschaftstabelle "Rechner_Properties" lassen sich zusätzliche DNS-Funktionen, wie z.B. Aliases, definieren oder im "TXT"-Feld Einträge zum Standort zu generieren sowie im Feld "HINFO" Angaben zur Hardware zu hinterlegen. Weitere Aufgaben des Domain Name Service z.B. für die Verwaltung und Verteilung öffentlicher Schlüssel zur Authentifizierung der eingesetzten Rechner lassen sich nun problemlos umsetzen und mit geringem Aufwand administrieren.

Kapitel 17

Datenbankgestützte Überwachung

17.1 Generelle Gedanken

Die Verfügbarkeit der Rechnerinformationen in einer Datenbank erlaubt nun auch die Automatisierung von Überwachungsfunktionen. Dadurch läßt sich der Aufwand senken, der üblicherweise hier erforderlich ist. Nun muss man sich nicht mehr auf die "Autodetect"-Funktionen verlassen, die einige der Netzwerktools mitbringen. Diese "übersehen" im Zeitpunkt der Abfrage z.B. nicht eingeschaltete oder nicht funktionsfähige Arbeitsstationen, Server und Netzwerkkomponenten. Darüberhinaus bieten sie nicht immer die Möglichkeit, Maschinen als "defekt", "in Wartung" etc. zu markieren.
Die folgenden Abschnitte werden sich allgemein oder am Beispiel bestimmeter Überwachungstools mit den Möglichkeiten der datenbankgestützten Überwachung auseinandersetzen. Vorher wird mit dem Simple Network Management Protocol (SNMP) ein wichtiges Werkzeug zur Beobachtung des Zustandes von Netzwerkkomponenten und Rechnern besprochen.
In diesem Abschnitt kann es jedoch nicht darum gehen, die generelle Eignung einzelner Werkzeuge für die Systemüberwachung in alle Details zu erläutern. Dazu gehören u.a. Fragen der Performance, der plattformübergreifenden Einsetzbarkeit, der Anforderungen an die Hardwareausstattung. Eher wird die Fragestellung zu erörtern sein, wie gut einzelne Tools mit den durch die Datenbank vorgegebenen Strukturen zu konfigurieren sind, wie leicht sie sich an die Vorgaben anpassen, bzw. ihre Konfigurationsdateien generieren lassen. In einigen Fällen wird es sinnvoll sein, zusätzliche Datenfelder einzufügen, um bestimmte Aspekte der Überwachung besser

berücksichtigen zu können. Hierzu zählt z.B. eine Zeitkomponente, welche festlegt, wann es überhaupt sinnvoll ist, die Funktion einzelner Maschinen zu überprüfen. So kann verhindert werden, dass Ausfallmeldungen über regulär abgeschaltete oder sich in Wartung befindliche Maschinen generiert werden.
Bei großen Beständen verteilter Arbeitsplätze kann es weiterhin sinnvoll werden, eine örtliche Positionsangabe mit dem Einzelgerät zu verknüpfen und dieses in den Überwachungstools geeignet auszuwerten. Diese Fragestellung bezieht auch Information bestimmter Personen bzw. Personengruppen ein, die in kritischen Situationen oder in Fehlerfällen automatisch benachrichtigt werden sollen. Diese Verknüpfungen erhöhen zwangsläufig die Komplexität des Managementsystems, lassen sich aber nahtlos in die Struktur der Datenbanktabellen einfügen.
In den nächsten Absätzen wird es deshalb darum gehen einige populäre Überwachungstools vorzustellen und auf ihre Eignung in der Zusammenarbeit mit anderen Werkzeugen des hier vorgestellten Projektes zu erörtern. Dazu zählt weiterhin ein kurzer Einblick in die realisierbaren Systemüberwachungstechniken, wobei das Simple Network Management Protocol seit längerem eines der Standardtechniken repräsentiert. Dieses wird üblicherweise ergänzt durch Techniken, die Erreichbarkeit einzelner Services durch automatische Anfragen abzuprüfen, wobei der Ping-Test[1] auf Erreichbarkeit zu den einfachsten Mitteln der Rechnerüberwachung zählt.

17.2 Varianten der Systemüberwachung

Mit der Ausbreitung von lokalen Netzwerken und der Etablierung des Internets entsteht die Notwendigkeit sich einen schnellen und umfassenden Überblick über den Zustand eines Netzes verschaffen zu können. Deshalb existieren inzwischen eine große Auswahl an Programmen und einige teilweise konkurrierende Protokolle zur Überwachung von Maschinen und Netzwerkkomponenten. Im vorgestellten Projekt kommen jene Tools zum Einsatz, welche umfassend für die Zielplattform implementiert und frei verfügbar sind.
Das SNMP-Protokoll erlaubt die schnelle und einfache Übertragung von Datenpaketen zur Zustandsbeschreibung von Netzwerkgeräten, wozu die in den vorhergehenden Abschnitten vorgestellten und einige weitere Variablen zählen. Jedoch genügt es in einigen Fällen nicht festzustellen, dass ein bestimmter Dienst läuft, indem man überprüft, ob der entsprechende Prozess in der Liste laufender Programme auftaucht. Es geht nicht darum, eine

[1] Das Programm **ping** ist die einfachste Form des Erreichbarkeitstests im Internet, da es auf dem Protokoll ICMP basiert, welches direkt in der IP-Schicht implementiert ist.

Innenansicht einer Maschine festzustellen, sondern den Zustand zu überprüfen, wie er sich aus Sicht des entfernten Benutzers darstellt. Vielfach ist erst dann eine Aussage sinnvoll, wenn ein Testzugriff über das Netzwerk erfolgreich durchgeführt wurde. Für diese Art der Überprüfung bieten sich Web-, FTP- und Mailserver sowie Authentifizierungsdienste an, da nur so eine Nutzersicht emuliert werden kann.
Die Auswertung der gewonnenen Informationen über eine Netzwerkmaschine und ihre Dienste obliegt dem entsprechenden Netzwerküberwachungstool, welches diese Fähigkeiten eventuell parallel zur beschriebenen SNMP-Schnittstelle implementiert. Eine einfache Basiskontrolle erfolgt üblicherweise mittels des Ping-Tests, welcher in fast allen entsprechenden Applikationen vorhanden ist.

17.3 "Uptime"-Überwachung

Neben Meldungen über den augenblicklichen Zustand eines Netzwerkes möchte man sich auch einen Überblick über Trends und Zeitreihen verschaffen, die einen tieferen Einblick in die Funktion eines Netzwerkes geben. So ermitteln reine Erreichbarkeitstests mittels Ping zwar, ob eine Maschine erfolgreich den Betriebssystemkern laden konnte, sagen aber nichts darüber aus, ob dieses Gerät auch im Benutzerbetrieb zur Verfügung steht. Hier kann die Überwachung der SNMP-Variable zur "Uptime", einem Zähler zur Länge der Betriebszeit der Maschine seit dem Booten, ansetzen.
Ein PHP-Skript bzw. Perl-CGI[2] kann für eine webbasierte Darstellung in einer Tabelle oder Ähnlichem anhand der Datenbankfelder einfach erstellt werden. Der Overhead eines solchen Tools fällt relativ klein aus und erfordert keine besonders hohen Systemresourcen. Darüberhinaus eignet es sich für plattformübergreifende Anzeigen, die zudem noch orts- und maschinenunabhängig sind.

17.4 SNMP

SNMP steht für Simple Network Management Protocol und wurde als Protokoll für Netzmanagement-Dienste in TCP/IP-Netzen entwickelt. Die Überwachung und Administration von LAN-Komponen-ten, wie z.B. Switches, Routern und Hubs in heterogenen Netzen auf Basis des TCP/IP-Protokolls war ursprünglich die einzige Aufgabe von SNMP. Inzwischen hat sich der Anwendungsbereich von SNMP um System- und Application-Management erweitert.

[2] Common Gateway Interface, welches im HTML-Standard definiert ist

Ähnlich wie bei TCP/IP, wo der Begriff nicht nur die Protokolle als solche, sondern das gesamte entsprechende Netzwerk bezeichnet, steht auch SNMP nicht nur für das Protokoll allein, sondern für das gesamte entsprechende Management-System. Wie bei TCP/IP üblich, werden auch die für SNMP relevanten Dokumente von der IAB (Internet Architecture Board) in RFCs (Request for Comment) hinterlegt[3].
Das Simple Network Management Protocol schafft eine allgemeine Abstraktionsebene, um plattform- und geräteunanbhängig Betriebsparamter zur Verfügung zu stellen. Grundlage für das Management der zu verwaltenden Komponenten ist die genaue Beschreibung der zu administrierenden Bestandteile (Objekte) dieser Komponenten in der MIB (Management Information Base). Die MIB ist das informationstechnische Rückgrat eines jeden Network Management Agents. Sie enthält Informationen zu Eigenschaften, z.B. zu Name, Typ, Syntax, Zugriffsrechten und Status jeder einzelnen Komponente. Für viele Hard- und Software-Komponenten werden vom Hersteller eigene MIB's mitgeliefert, die mit geringem Aufwand individuellen Bedürfnissen angepaßt werden können. Die Codierung der MIB erfolgt in ASN.1 (Abstract Syntax Notation One). ASN.1[4] wurde auch von ISO als Standard für den Präsentationsschicht[5] genormt.

17.4.1 Implementierungen unter Linux

Unter Linux stehen im wesentlichen zwei Implementierungen des SNMP zur Verfügung. Die ältere wurde an der Carnegie-Mellon-University entwickelt und firmiert unter der Bezeichnung CMU-SNMP. Die neuere und erweiterbare Version stammt von der University of California in Davis[6]. Darüberhinaus gibt es eine ganze Reihe von SNMP-Clients, welche eigene Implementierungen mitbringen oder auf die Bibliotheken der beiden genannten Pakete zurückgreifen. Ein Beispiel ist die Tcl/Tk-basierte Netzwerküberwachungssoftware "tkined", welche auf der Bibliothek "scotty" basiert.
SNMP bildet in den meisten Fällen die Basis, auf der viele Programme und

[3]Die grundlegenden RFCs für SNMPv1 (Version 1) sind:

- RFC1155 : "Structure and Identification of Management Information for TCP/IP-based Internets (SMI)", Mai 1990
- RFC1157 : "A Simple Network Management Protocol (SNMP)", Mai 1990
- RFC1212 : "Concise MIB Definitions", März 1991
- RFC1213 : "Management Information Base for Network Management of TCP/IP-based Internets: MIB", März 1991

[4]siehe hierzu ISO/IEC 8824 und 8825
[5]sechste Schicht im siebenschichtigen OSI-Referenzmodell
[6]http://www.ucdavis.edu

Überwachungswerkzeuge aufsetzen. Nach den eingangs angestellten Überlegungen wird man vielfach SNMP nicht ausschließlich einsetzen, sondern seine Funktionalität durch weitere Test-Tools ergänzen. Deshalb erfolgt die Vorstellung einiger verfügbarer Anwendungen in eigenen Abschnitten.

17.4.2 Funktionalität des UCD-SNMP-Pakets

Das UCD-SNMP-Paket stellt die meisten der üblichen Host-Variablen unter Linux zur Verfügung. Dazu gehören die Standard-Location-Parameter, die Netzwerkinformationen zu Interfaces, ARP[7], ICMP, TCP, inklusive der Anzahl der transferierten Pakete in verschiedenen Kategorien.
Eine ganze Reihe weiterer Zustandsdaten wird über zusätzliche SNMP-Module abgedeckt. Hierzu gehören Angaben zur Auslastung des Filesystems, die Überwachung der Anzahl bestimmter Prozesse und die Möglichkeit der Warnung bei Überschreitungen von Grenzwerten. Es läßt sich die Zahl der angemeldeten Benutzer, die durchschnittliche Auslastung der Maschine und Informationen über die installierten Softwareprodukte gewinnen.

17.4.3 Erweiterbarkeit des UCD-SNMP

Neben den beschriebenen Standardschnittstellen ist es durch Erweiterungen des Tools vorstellbar, externe Programme zur Gewinnung von Informationen ausführen zu lassen und deren Ausgaben bzw. Rückgabewerte zu interpretieren. Dadurch kann es sehr leicht gelingen die Generierung bestimmter Systeminformationen zu testen und später bei Bedarf direkt im Programm zu implementieren. Weiterhin lassen sich auf diesem Wege die Ausgabe von anderen Programmen auf eine einheitliche Schnittstelle bringen.
Die Erweiterung der Funktionalität wurde im vorliegenden Projekt dazu benutzt, Informationen des System Management Busses zu CPU-Temperatur und Drehzahl der verschiedenen Lüfter ausgenutzt. Eine ganze Reihe weiterer Datenerhebungen ließe sich auf diesem Wege bewerkstelligen. Jedoch gibt es einige Einschränkungen bezüglich der verwendeten Datenfelder (nur String-Variablen oder Return-Codes bis zum maximalen Wert von 255 sind erlaubt), so dass nach einer Testphase über eine feste Einbindung als Modul nachgedacht werden sollte.

[7] Address Resolution Protocol

17.5 Momentaufnahmen des Netzwerkzustandes

Eine der Hauptaufgaben der Netzwerkadministration liegt darin, sich zu jeder Zeit einen schnellen Überblick über den Zustand des gemanagten Netzes verschaffen zu können. Bei Ausfällen einzelner Komponenten oder Teilbereichen des Netzwerkes sollte die Systemadministration bei Anfragen und Beschwerden von Benutzern zumindest bereits grob vorab informiert sein.
Ein Beispiel für eine solche Momentaufnahme bietet die Benutzung des Werkzeuges **tkined**. Dieses setzt auf der Skriptprogrammiersprache Tcl/Tk auf, verfügt über eine grafische Oberfäche und läßt sich gut auf verschiedene Plattformen portieren. Zur Ermittlung von Systemzuständen setzt es auf eine eigene SNMP-Implementierung, der "Scotty"-Bibliothek auf. Die Benutzerschnittstelle verfügt über eine ganze Reihe von Optionen, Netzwerkkomponenten darzustellen, diese in Gruppen zusammenzufassen und Abgrenzungen vorzunehmen. Es lassen sich eine größere Menge von Daten unterhalb eines Symbols zusammenfassen und bei Bedarf eine Detailansicht generieren. Die Konfigurationsdatei dieses Tools erlaubt eine Erstellung mittels eines Perlskriptes, welches die notwendigen Datenbankinformationen beschafft. Im Anhang (F.3.1) erfolgt der Abdruck eines Beispiel-Skriptes (**mk_tkined**), das Einblicke in einige Möglichkeiten dieses Tools zur Rechnerüberwachung bietet. Dazu zählen der einfache Pingtest bis zum Monitoring der Festplattenbelegung, Last auf den Ethernet-Interfaces, Zahl der eingeloggten Benutzer und Zustände des System Management Busses. Informationen, welche sich nicht besonders gut in Diagrammen visualisieren lassen oder als Floating-Point-Variablen vorliegen, eignen sich jedoch kaum zur Anzeige in **tkined**.
In Abhängigkeit der Zahl der zu überwachenden Systeme bzw. Einzelaspekte wird zu überlegen sein, welche Darstellung man anstrebt: Ausgabe in einem einzigen Fenster, oder nach Gruppen verteilt, über mehrere. Als Anzeigemaschine kommt ein dediziertes Gerät in Frage, eine Abfrage an beliebigen Arbeitsstationen kann nur über die Realisierung des X11-Protokolls erfolgen. Weitere Schwierigkeiten sind in der Performance bei sehr hohen Rechnerzahlen oder vielen zu überwachenden Diensten zu erwarten. Abstriche in der Ergonomie der Anzeige müssen bei größeren Datenmengen hingenommen werden, da die zur Verfügung stehenden Einstellungen des Programmes beschränkt sind. Diese Frage stellt sich beim ähnlich orientierten Tool "cheops" in etwas anderer Weise: Es basiert zwar auf einer deutlich performanteren Grafikbibliothek, dem Gimp-Toolkit, allerdings sind einige Abstriche am Benutzerinterface und der Funktionalität hinzunehmen. Jedoch eignet sich die Konfigurationsdatei von **cheops(-ng)** gleichfalls zur

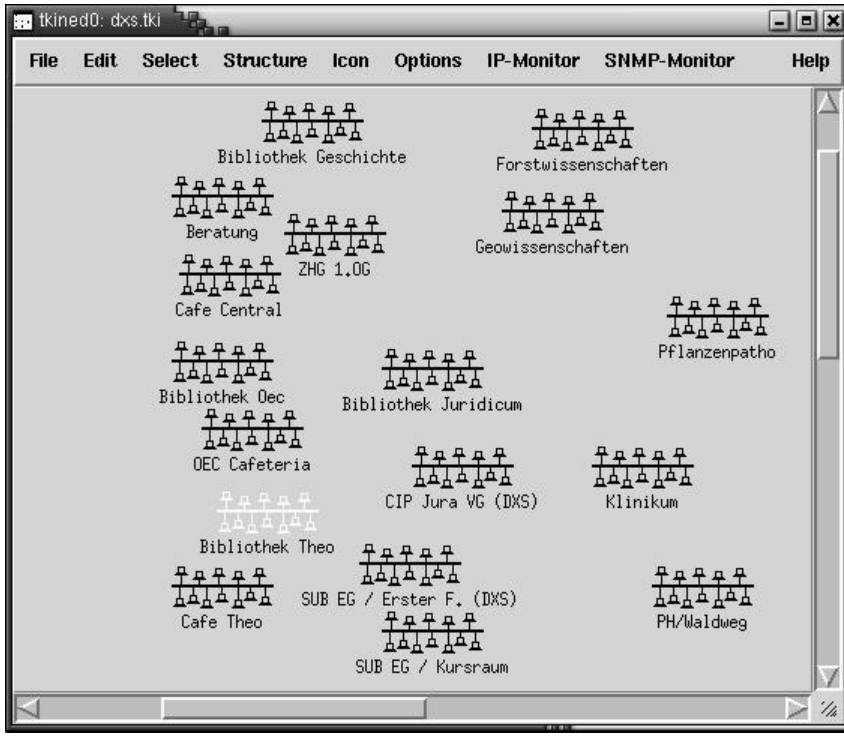


Abbildung 17.1: TKined-Fenster im Überblicksmodus

Skriptgenerierung.

17.6 Netzwerküberwachung mit NetSaint

17.6.1 Das Tool

NetSaint verfügt über eine modulare Struktur indem das Frontend zur Anzeige der gesammelten Daten vom Kollektor zur Erstellung des Überblicks komplett unabhängig agiert. Beide Teile kommunizieren über eine zentrale Log-Datei oder bei höheren Ansprüchen mittels SQL-Datenbank miteinander. NetSaint erfreut sich inzwischen einiger Beliebtheit, so dass es als Paket bei der aktuellen SuSE-Distribution mitinstalliert werden kann.
Aus Sicht dieser Arbeit liegt der interessanteste Aspekt von NetSaint in dessen vorteilhafter Konzeption in Bezug auf das Zusammenwirken mit der vorgestellten Struktur des Rechnernetzwerkes und seiner Abbildung in der

Datenbank. NetSaint bietet nicht nur Ad-Hoc-Ansichten eines Netzwerkes, sondern kann längerfristige Statistiken und Trends darstellen. Das erlaubt kurzfristige Fehler und Ausfälle von Maschinen und Netzwerkteilen auszublenden und einen Gesamtüberblick zu erhalten. Systematische Probleme können so besser erkannt und behoben werden.

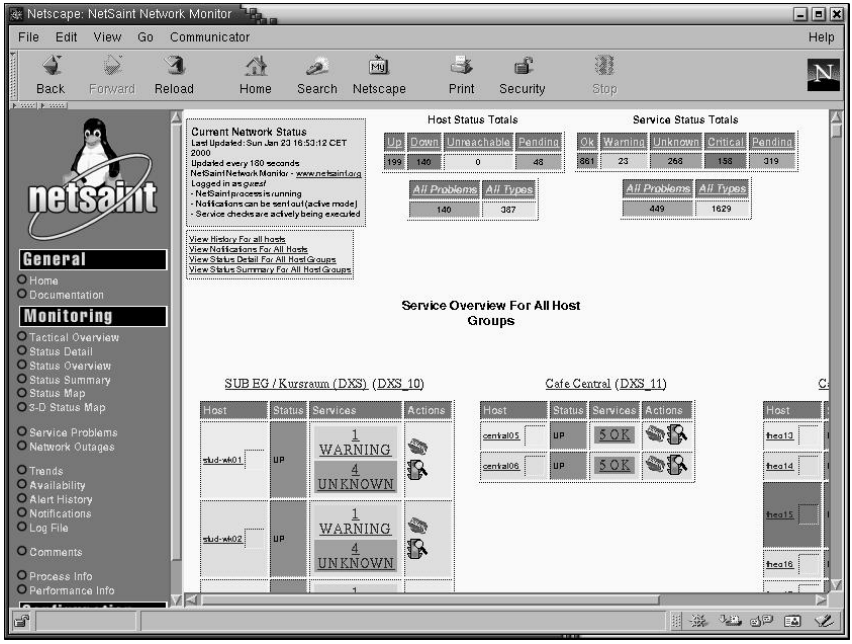


Abbildung 17.2: NetSaint: Rechner-Überblick

Eine ganze Reihe von Konfigurationsdateien, welche sich üblicherweise im Verzeichnis */etc/netsaint* wiederfinden, steuern den Ablauf von NetSaint und erlauben vielfältige Anpassungen des Verhaltens. Ein Teil dieser Konfigurationsdateien wird mittels Perlskript analog zu den bisher beschriebenen Werkzeugen aus der Datenbank generiert. Dies betrifft in erster Linie die Definition der zu überwachenden Maschinen und deren Dienste, sowie einige Aspekte des Frontends. Nähere Ausführungen sowie das Beispielskript können dem Anhang (Abschnitt F.3.2) entnommen werden. Die ausführliche Beschreibung der verschiedenen Funktionen sowie Hinweise zu den unterschiedlichen Plugins und Erweiterungen finden sich auf der Homepage[8] des Projektes.

[8] Siehe hierzu [W5]

17.6.2 Überwachungs-Backend

Das Backend von Netsaint ist aus Performancegründen in der Programmiersprache C geschrieben und bietet Schnittstellen, die das Einbinden zusätzlicher Plugins erlauben, welche wiederum als C-Programm, Perl- oder Shellskript vorliegen können. Eine Reihe von Programmierern haben so das Spektrum der Überwachungsaufgaben ausgedehnt, so dass inzwischen eine breite Palette zur Kontrolle diverser Internet-Services, wie WWW, FTP, Mail (mit allen beteiligeten Protokollen) etc. zur Verfügung stehen. Diese Überprüfung geschieht nicht auf der Servermaschine selbst, sondern erfolgt vom Systemmonitor aus, womit ein besseres Bild des (Nicht-)funktionierens aus Client-Sicht generiert wird.
Darüberhinaus können RPC-Dienste[9] abgefragt werden und es existiert eine gute SNMP-Implemen-tierung, womit die eingangs beschriebenen Möglichkeiten des SNMP nahtlos eingegliedert werden können. Eine ausführlichere Beschreibung zur Konfiguration der Kommandos und deren Flexibilität sei dem Anhang F.3.2 entnommen. Die Konfiguration des Systemmonitorings erfolgt gestuft; es lassen sich zeitliche Komponenten, hierarchische Abhängigkeiten von Rechnern und Diensten sowie Meldungen zu Ausfällen erzeugen. Bestimmte Zusatzfunktionalitäten erfordern evtl. eine Erweiterung der in der Datenbank zu einzelnen Systemen gespeicherten Informationen.

17.6.3 Webfrontend

Die Argumente, welche für eine Benutzung einer Webschnittstelle zur Anzeige und Überblicksdar-stellung von Daten sprechen, wurden ausführlich bereits im Abschnitt 15 dargelegt. Das Gesagte gilt weiterhin für die Performanceüberlegungen einer solchen Schnittstelle und ihren Darstellungswerkzeugen, den Webbrowsern.
NetSaint liefert eine umfangreiche Liste an Darstellungen für die erhobenen Daten. Diese reicht von einem Überblick über den Zustand der Rechner und Rechnergruppen, sowie die Verfügbarkeit ihrer jeweiligen Services bis hin zu den Detaillisten zu deren Einzeldarstellung. Auf diese Weise kann der Abstraktionsgrad je nach Erfordernis gewählt werden.
Das Frontend bietet weiterhin die Möglichkeit eine Statuskarte zu zeichnen, worin die einzelnen Maschinen, so ihre Position bekannt ist und aus der Datenbank zur Erstellung der Konfigurationsdatei ermittelt werden konnte, in ihrer Lage zueinander verzeichnet sind. Das erleichtert den Verantwortlichen vor Ort bzw. der Hardwareadministration ein leichteres Auffinden der Maschine.

[9] Remote Procedure Call

Abbildung 17.3: NetSaint: Status-Überblick

Kapitel 18

Inventarisierung und Bestandsanalyse

18.1 Betriebswirtschaftliche Anforderungen

Außerhalb der rein privaten Nutzung von Rechnern und Netzwerken stellen sich überlicherweise einige zusätzliche Bedingungen. Dazu gehört der Nachweis des Verbleibs bzw. des aktuellen Einsatzes der Betriebsmittel sowie ihre wertmäßige Erfassung. In größeren Abteilungen bzw. aus Sicht eines Unternehmens, einer Organisation etc. lassen sich diese Anforderungen nur unter hoher Aufwand erfüllen. Dadurch entstehen schnell eine Reihe von Problemen, die meistens zu höherem Arbeitsaufwand und Kosten führen.
Unter anderem für diese Aufgaben läßt sich die eingeführte Datenbank nutzen. So läßt sich der Eingang bzw. Abgang von Rechnern und Komponenten lückenlos verfolgen und deren physikalischer Verbleib nachvollziehen. Das kann das Verfahren im Garantiefall, bei Entscheidung über Neubeschaffungen bzw. Updates erleichtern. Durch die Verknüpfung der Daten zum Betrieb und zur Systemkonfiguration mit dem Wissen über Beschaffung, Lieferanten etc. bleiben die Informationen leichter konsistent. Die verschiedenen Frontends zur Steuerung der Datenbank wurden an früherer Stelle (siehe hierzu Abschnitt 15) unter Berücksichtigung verschiedener Benutzergruppen und ihren Bedürfnissen bereits vorgestellt.
Im weiteren wird es darum gehen, welche Werkzeuge dem Systemadministrator an die Hand gegeben werden können, um einen Überblick über die eingesetzte Hardware zu behalten. Wichtig hierbei sind Fragen der einfachen Kennzeichnung und Lesbarkeit, sowie die Aufgaben, welche sich aus der Pflege der Datenbestände ergeben.

18.2 Inventarisierung

Bei Erfassung der Bestände gibt es einige Tatbestände, die in die Berücksichtigung einbezogen wurden. Die Einzelteile sollten leicht zu identifizieren sein. Gleich aussehende oder sehr ähnliche Komponenten, wie CPU, Speicher, Adapterkarten müssen sich voneinander unterscheiden lassen, da sie trotz ihrer evtl. Gleichheit nicht vom selben Lieferanten stammen müssen oder zum gleichen Zeitpunkt bestellt wurden. Weiterhin wird man nicht davon ausgehen können, dass ein Arbeitsplatzsystem immer in der gleichen Konfiguration betrieben wird. Es können notwendige Erweiterungen hinzukommen oder Einzelkomponenten wegen Defektes oder Garantieanspruches ausgetauscht werden. Bei einer großen Anzahl von Rechnern in einem Netzwerk benötigt man zur schnelleren Identifikation von Einzelkomponenten und ganzen Systemen eigene Verfahren zur Kennzeichnung, die über die Herstelleraufdrucke hinausgehen.
Deshalb bietet sich ein Aufkleber mit einem Strichcode an. In vielen Fällen sind Komponenten bereits durch solche[1] gekennzeichnet, jedoch fällt diese Bezeichnung meistens zu unterschiedlich aus, um den hier skizzierten Anforderungen zu genügen. Mit einem zu diesem Zweck generierten Barcode, der zusätzlich angebracht wird, fällt die Unterscheidung am leichtesten. Dieser Code läßt sich von einem entsprechenden Lesegerät identifizieren. Für die Fälle, dass ein solches Gerät nicht zur Verfügung steht, fügt man die Entsprechung des Codes als Zahl hinzu.
Lesegeräte für Barcodes gibt es in verschiedenen Ausführungen, dasselbe gilt für die standardisierten Codes. Üblicherweise wird dieser Barcodescanner in den Tastaturanschluß eingeschleift, so dass eine Datenerfassung über den Strichcodeleser wie der Input über die Tastatur funktioniert. Damit entfallen aufwändige Anpassungen der Software nach Eingabemethode. Stehen nur einige Lesegeräte zur Verfügung, plaziert man diese an Stellen häufiger Benutzung, also dem höchsten Einsparpotenzial an Zeit. An Orten sporadischer Eingabe genügt die Tastatur.
Als Schnittstelle zur Datenbank sind verschiedene Frontends vorstellbar: Eine skriptbasierte Eingabe eignet sich bei umfangreichen sich wiederholenden Beständen, zum Beispiel der Generierung von Datenbankeinträgen aus der *dhcpd.conf* bei Neueinführung einer Datenbank in bereits bestehenden Netzwerken. Ein Beispielskript hierfür findet sich in Anhang D.5. Für den täglichen Umgang sind ein grafisches Frontend bzw. ein Webfrontend vorstellbar, wobei letzteres den Vorteil der größeren Plattformunabhängigkeit hat. Einige professionelle Datenbanken bringen konfigurierbare Eingabemasken mit,

[1] Neben den Herstellerkennzeichnungen versehen Händler ihre Ware mit eigenen Markierungen.

was bei der Auswahl mit beachtet werden sollte.
Weiterhin sollten auch die Arbeitsplatzrechner gut sichtbar gekennzeichnet sein, was dem Administrator und dem Benutzer die Identifikation erleichtern. Auch hier bieten sich wieder selbstklebende Papieretiketten an, da diese leicht bedruckbar und im Fall eines Gerätetausches leicht neu zu generieren sind.

Abbildung 18.1: Rechnerlabel

An dieser Stelle wurde ein Tool in Perl erstellt, welches sich wiederum der Datenbank bedient um die entsprechenden Informationen auf Etiketten bringen zu können. **label** erzeugt einen Postscript-Code, welcher entweder direkt an den Drucker oder über geeignete Konvertierer ausgegeben werden kann. Dadurch werden zwischengeschaltete Formatierungswerkzeuge überflüssig. Die verschiedenen Kommandozeilenoptionen und die genaue Syntax beim Aufruf des Tools sind im Anhang angegeben.

18.3 Bestandsaufnahme

18.3.1 Erzeugen von Datenlisten

Über die Konfiguration der einzelnen Clients und automatisch installierten Server im Netz entscheiden die entsprechenden Software-Daten und Hardwareeinträge in der Datenbank. Um sich ein schnelles Bild zu verschaffen, liest das Tool **report** die Datenbank aus und bereitet die Ausgaben für den Systemadministrator auf. Zum einen kann eine Teileliste als Postscriptdokument erzeugt werden, welche zur Inventarisierung einmal abgeheftet und einmal im System selbst hinterlegt werden kann.
Darüberhinaus lassen sich auch einfache Konfigurations- und Hardwarelisten generieren, die einen leichten Überblick über ein System geben. Diese

eignen sich zur Ordnerablage (so gewünscht) oder zur Hinterlegung in der Maschine selbst, um dem Servicepersonal vor Ort wichtige Daten an die Hand zu geben.

18.3.2 Verschiedene Exportmöglichkeiten

Sollen in regelmäßigen Abständen Listen einer bestimmten Ausrichtung generiert werden, ohne dabei Bestände in der Datenbank zu modifizieren, bietet sich neben der ODBC-Schnittstelle der Datenexport in Form von Excel-Tabellen-Dateien an, die über geeignetes Perlmodul[2] erstellt werden können. Weiterhin wären Schnittstellen zu anderer betriebswirtschaftlicher Standardsoftware, wie SAP-R/3, denkbar.
Stehen keine Standardschnittstellen für einen direkten Import von Daten zur Verfügung, bietet eine Datenbank immer noch die Option eines einfachen Auslesens mit den Kommandozeilentools, womit sich kommaseparierte Listen[3] erstellen lassen, die sich leicht an anderen Stellen wieder importieren lassen.

[2] beschreiben z.B. in der iX 6/2001
[3] andere Trennzeichen sind natürlich alternativ vorstellbar

Rechner: 204 s11

Inventurliste

Internet-AG - Platz der Goettinger Sieben 5 - 37073 Goettingen

s11
stud.uni-goettingen.de
134.76.60.31
204
Inventarliste Hardware
Internet-AG
1 340760 600317
20.6.2001 <hotline@stud.uni-goettingen.de>

20.6.2001

Internet-AG
Universitaet Goettingen
Platz der Goettinger Sieben 5
Telefon: 39-8392

Produktname (Produktart)	Beschafft	Garantie_bis	Seriennummer (intern)	Barcode	Brutto-Preis
UDMA 100 IDE-Contr. 2 Anschl., Promise-Chip (Adapter-Karte)	2001-03-28	2002-03-27	121473		121.80 DM
MidiTower ATX 300W (ATX-Gehaeuse)	1998-07-20	1999-07-19	013883		1.00 DM
GA-6BXD Dual, i440BX Slot1 (ATX-Mainboard)	1998-01-01	1999-01-01	023753		0.00 DM
Medalist 4310, ST34310A, 4,3GB (Festplatte)	1999-05-19	2002-05-18	120602		198.00 DM
Medalist 4310, ST34310A, 4,3GB (Festplatte)	1999-05-19	2002-05-18	120643		198.00 DM
Barracuda ATA III, ST320414A, 20GB (Festplatte)	2001-03-02	2004-03-01	121121		342.20 DM
Barracuda ATA III, ST320414A, 20GB (Festplatte)	2001-03-02	2004-03-01	121111		342.20 DM
1,44MB FD Alps (Floppy-Laufwerk)	1999-02-01	2000-01-31	073383		25.00 DM
S3 Trio64(X) (Graphikkarte)	1999-10-15	2000-10-14	060923		30.00 DM
Clones 100Mbit, 21143Chip, WoL (Netzwerkkarte)	2001-03-28	2002-03-27	053833		80.04 DM
Clones 100Mbit, 21143Chip, WoL (Netzwerkkarte)	2001-03-28	2002-03-27	053813		80.04 DM
Celeron 500 PPGA (Prozessor)	2000-04-17	2001-04-16	032593		231.00 DM
256MB SDRam, PC100, UC16M08 (Speicher)	2000-04-04	2001-04-03	043712		426.50 DM
256MB SDRam, PC100, UC16M08 (Speicher)	2000-04-04	2001-04-03	043722		426.50 DM
256MB SDRam, PC100, UC16M08 (Speicher)	2000-04-04	2001-04-03	043733		426.50 DM
IBM alt PS2-Stecker (Tastatur)	1997-01-01	1998-01-01	093412		1.00 DM
			Gesamtpreis:		**2929.78 DM**

Abbildung 18.2: Beispiel Postscript-Inventarliste

Teil IV

Fazit

Kapitel 19

Schlußfolgerungen

Die hier vorgestellte Lösung zum Betrieb größerer Rechnerpools läßt sich durch zwei für die Entwicklung zentrale Gesichtspunkte kennzeichnen:

- Zum einen sollte das Profil der Lösung in verschiedener Hinsicht offen sein. Dies gilt sowohl für die Integration unterschiedlicher Rechnergenerationen und speziellerer Hardware, als auch für die Offenheit der Lösung für vielfältige Applikationen und auch Betriebssysteme. Von Anforderungsseite her gedacht, war die Öffnung des Systems Kernbedingung.
- Zum anderen ging es bei der Entwicklung darum, die Administration möglichst einfach, d.h. auch denkbar kostengünstig zu gestalten, ohne dabei wichtige Aspekte wie die Sicherheit des Systems zu vernachlässigen.

Im Mittelpunkt des Lösungskonzepts steht letztlich die Administration des Rechnernetzwerks über die Datenbank. Diese erlaubt zugleich ein hohes Maß an Standardisierung als auch eine Flexibilisierung im Einzelfall.
An dieser Stelle sollen nochmals die zentralen Momente des Administrationskonzepts zusammengefaßt werden. Im folgenden letzten Kapitel 20 können dann abschließend einige Erweiterungsmöglich-keiten und speziellere Probleme abgehandelt werden, die nicht zum Kern der Problemstellung gehören.
Die entwickelte Datenbank bietet eine offene Beispielimplementation, die in verschiedene Richtungen weiterentwickelt werden kann und auf unterschiedliche Problembereiche übertragbar ist. Sie geht zunächst von der Aufgabe aus, für eine Anzahl recht ähnlicher Arbeitsplätze den Administrations- und Installationsaufwand zu reduzieren. Mit Thin-Clients auf Linux-Basis wird hierfür eine neue Rechnergeneration vorgestellt, die die zentralen Anforde-

rungen nach Arbeitsplatzergonomie und der Reduktion laufender Kosten erfüllen kann.
Thin-Clients erlauben zunächst auf Hardware-Seite den Wert von Investitionen länger zu erhalten. Die klassischen betrieblichen Forderungen nach einer bestandsmäßigen Aufnahme und wertmäßigen Behandlung der EDV-Investitionen lassen sich über wenige zusätzliche Datenbankfunktionen in die Administration einbeziehen. Die hier präsentierte direkte Verknüpfung in einer einzigen Datenbank stellt dabei eine beispielhafte Anwendung dar, genauso ließen sich Schnittstellen zu bereits bestehenden Systemen denken. Da diese auf Standardprotokollen beruhen, gelingt es mit geringem Aufwand auch administrationsfernere Abteilungen wie Verwaltungen anzubinden und sich einen Überblick über betriebswirtschaftliche Variablen verschaffen zu lassen. Hierbei geht es also meistens nicht um die komplette Neuimplementation von Aufgaben, sondern um die bessere Integration in schon bestehende Strukturen. Die gewählte Aufgliederung nach Einzelteilen oder Komponentengruppen orientiert sich dabei an der klassischen Beschaffungssituation. Zur späteren Identifizierung und Unterscheidung der Komponenten und ihrer wertmäßigen Verwendung läßt sich eine solche Struktur nur schwer vereinfachen.
Die Thin-Clients erfordern zur Installation einen gegenüber der klassischen Workstation erhöhten Einrichtungsaufwand, der sich jedoch bei einer schon geringen Zahl gleichartiger Anforderungen bezahlt macht. Die hier präsentierte Lösung hatte bei ihrer Implementierung die spätere Skalierung im Blick: Ausgehend auf der einmal geschaffenen Basis läßt sich mit nur geringem Mehraufwand, der weit unter dem für die einzelne klassische Workstation liegt, ein weit höhere Zahl als die eingesetzten 400 Geräte[1] verwalten. Bei den meisten hier beispielhaft beschriebenen Tools spielt die Zahl der tatsächlich verwalteten Systeme keine Rolle. Hier liegen die Beschränkungen eher in den Möglichkeiten der betrieblichen Organisation, der Leistungsfähigkeit des Netzwerkes und der gegebenen Anforderungen.
Vereinfachungen der Administration zeigen sich über die Installation hinaus im laufenden Betrieb auch durch die Realisierung der netzbasierten Software-Konfiguration. Im Spektrum der Lösungen von Thin-Clients bis hin zur vollausgestatteten Workstation, liegt die vorgestellte Implementierung entgegengesetzt zum "klassischen" Arbeitsplatz-PC. Dieser Umsatzung liegt die Idee zugrunde, die Software auf Client-Seite so knapp und damit fehlerarm wie möglich zu halten. Da diese Software nur bereits lange definierte Internet-Protokolle umsetzt, ist hier von geringstem Update-Aufwand

[1] Dieses umfaßt die Zahl der direkt in diesem Projekt gemanagten Rechner. Zählt man die weiteren Umsetzungen in anderen Bereichen der Universität dazu, erhöht sich die Anzahl weiter.

auszugehen. Dies betrifft sowohl die laufende Veränderung von Software- und Netzkonfigurationen, als auch die Kontrolle und Sicherung des normalen Betriebs.
Lösungen, wie sie auch einige kommerzielle Hersteller solcher Thin-Clients einsetzen, die auf mechanikfreien Festspeichern beruhen, verlagern einigen Aufwand vom Server weg und erreichen eine gewisse Autonomie eines solchen Systems. Dadurch erhöht sich auf der anderen Seite jedoch der Kostenaufwand für die nun einzusetzende recht spezielle Hardware, die Anforderungen an die Ausstattung mit bestimmten Hardwareschnittstellen für die Clients und die Pflege des nun wieder lokal verfügbaren Filesystems. Im Zuge allfälliger Updates aufgrund von Sicherheitsanforderungen oder der Implementierung neuer Eigenschaften erhöht sich üblicherweise auch der Speicherplatzaufwand, der von der ursprünglichen Lösung vielleicht nicht mehr bereitgestellt wird, womit Auswirkungen auf den Gesamtaufwand verbunden sind.
Die offene Architektur der zugrundegelegten Software bietet eine Reihe von Schnittstellen zu anderen Plattformen und ist damit nicht genuin an bestimmte Anwendungen gekoppelt. Für die gesamte Aufgabenstellung kommen etablierte Standardprotokolle und Applikationen zur Anwendung, welche in den meisten Betriebsumgebungen bereits zur Verfügung stehen. Dabei wird auf wohldokumentierte Progammiersprachen, wie SQL und Perl zurückgegriffen, die nicht nur für das hier vorgestellte Linuxbetriebssystem zur Verfügung stehen.

Linux als ein Abkömmling der klassischen Unixbetriebssysteme, eingesetzt auf dem Personanlcomputer, bringt viele Erfahrungen und Systemdesigns mit, die einen Multiuser- und Multiprozessbetrieb unterstützen. Die Architektur der Benutzerrechte beim Zugriff auf das Dateisystem und zum Ausführen von Programmen sind Grundbedingungen des Rechnerbetriebes in modernen Netzwerken.
Der Einsatz freier Software bietet eine ganze Reihe von Vorteilen: Ein gesondertes Lizenzmanagement kann entfallen, auch wenn sich dieses sicherlich als Erweiterungsmodul der Datenbank zusätzlich realisieren ließe. Dadurch stellt aber die Skalierung der Zahl der Arbeitsplätze, zumindest was den Bereich der Fundamentalapplikationen anbetrifft, kein Problem dar. Dies betrifft sowohl die Thin-Clients, als auch die zur Verfügung zu stellenden Boot- und Applikationsserver. Wie bei den klassischen Lösungen bleibt jedoch die Korrellation von eingesetzter Software zum Ausbaustand der Hardware bestehen. Diese läßt sich jedoch durch die Verlängerung der Laufzeiten der Geräte z.B. als X-Terminal oder die gemeinsame Nutzung von Serverresourcen geringer halten.

Die Kombination von Thin-Clients und Datenbankadministration erlaubte uns schliesslich auch, klassische Aufgabenbereiche der EDV-Betreuung zu vereinfachen: Der Austausch defekter Geräte und Einzelteile geht reibungsloser vonstatten, da keine benutzerrelevanten Daten mehr an den Arbeitsplätzen vorgehalten werden. Die Ausfallzeiten lassen sich hierdurch stark verkürzen und die Vorhaltung von Ersatzteilen vereinfachen. Durch die Verknüpfung von wesentlichen Bestimmungsgrößen zur Ansteuerung der Hardware in der Datenbank entfällt die aufwändige und sich in vielen Fällen wiederholende Einzelkonfiguration der Maschinen. Dadurch ist es nun nicht mehr erforderlich, eine bestimmte Anzahl von Komponenten auf Vorrat zu halten, um auch im Fall von Defekten eine einheitliche Architektur vieler Systeme aufrecht zu erhalten, die bei klassischen Lösungen zumindest in Teilen die Konfiguration erleichterte. So reduzieren sich die Kosten der Lagerhaltung und die Gefahr, beim Ausfall eines Einzelteils ein ganzes Gerät außer Betrieb setzen zu müssen. Da die Administration gebündelt ist, lassen sich auch divergierende Hardware- Konfigurationen besser verwalten. Das Problem, bei Neubeschaffungen auf Standardkonfigurationen mit nur geringen Verfügbarkeitszeiträumen zurückgreifen zu müssen, entfällt. Die Pflicht zur sauberen Pflege der Datenbank, um funktionierende Gerätekonfigurationen zu erhalten, hat einen weiteren Vorteil in der besseren Verfolgbarkeit der Verwendung investierter Mittel.
Viele Anpassungen und Veränderungen der Standardlösungen für festplattenbetriebene Workstations lassen sich im Open-Source-Bereich einfach umsetzen, da nicht die Rechte der Programmierer bzw. Hersteller der Software verletzt werden. Weiterhin stehen wohldokumentierte Lösungen und Lösungsbeispiele von anderen im Internet bereits zur Verfügung. Dadurch entfällt zwar in vielen Fällen der direkte Zugriff auf die Support-Hotlines der verschiedenen Hersteller, was jedoch meistens durch das breite Angebot an Supportforen, Mailinglisten und Mitentwicklern mehr als überkompensiert wird. Viele Fragestellungen, die im Laufe dieser Arbeit auftraten, konnten im Bereich der Open-Source-Software schneller und einfacher geklärt werden, als im Kontakt zu den diversen kommerziellen Software-Anbietern.
Zuletzt kann durch die Einführung von Thin-Clients die Sicherheitsarchitektur verbessert werden: Da große Teile des Filesystems vom Server nur lesbar zur Verfügung gestellt werden, sind Manipulationen an wichtigen Binärdateien, wie Programmen und Bibliotheken erschwert. Weiterhin gelingt es nicht mehr so einfach, zusätzliche Programme, Log-Dateien oder Skripten zu verstecken, welche Systemleistung blockieren oder Angriffe versuchen. Der Umfang der zu überwachenden Filesysteme sinkt rapide, so dass sich auch aufwändigere Überwachungssysteme anbieten, ohne den Gesamtadministrationsaufwand erheblich zu erhöhen, da das Augenmerk auf den Servern

liegt, die in vielen Fällen bereits in Sicherheitsstrukturen eingebunden sind. In der vorgestellten Betriebslösung verschiebt sich die Sensibilität, was Sicherheit und Konfiguration anbetrifft, zum Server hin. Systemadministratoren müssen sich nun im klaren sein, dass Änderungen am Serverfilesystem oder die Verfügbarkeit von Applikationen sich über den Server hinaus auswirkt. Die Stabilität des gesamten Betriebes hängt nun in höherem Maße von den Servern ab. Dieses betrifft auch Erwägungen zur Systemsicherheit. Einbrüche auf die zentralen Maschinen haben nun weiterreichende Auswirkungen, bis hin zur Korrumpierung bzw. Unbrauchbarkeit der Clients. Die Einlußmöglichkeiten des "Superusers[2]"auf die Thin-Clients reduzieren sich durch die eingeschränkten Rechte im Dateisystem. Alle dauerhaften Änderungen müssen auf dem Server erfolgen. Die Benutzerrechte auf den Thin-Clients unterscheiden sich jedoch nicht von denen der Server bzw. klassischer Stand-Alone-Workstations.
Nicht jeder Arbeitsplatz läßt sich auf die beschriebene Weise administrieren. Rechner mit sehr speziellen Schnittstellen oder Erweiterungshardware oder besonderen Anforderungen sind nicht als Thin-Clients ausführbar. Hier bietet sich in vielen Fällen die automatische Installation und automatisches Update an, um den Administrationsaufwand zu senken. Für einige Bereiche spezieller Server führt jedoch weiterhin kein Weg an der manuellen Administration vorbei, wenn die Abbildung auf die vorgestellte Lösung zuviel Anpassungsaufwand erfordert. In diesen Grenzfällen heißt es weiterhin abwägen: Der Arbeitsaufwand einer separaten Administration steht dem Aufwand einer Einbettung in das datenbankgestützte System der Verwaltungs- und Konfigurationsskripten gegenüber.

[2] Systemadministrator, überlicherweise die UserID "root" mit der User- und Gruppenkennung "0"

Kapitel 20

Ausblick

Viele Aspekte der Desktop-Gestaltung, der eingehenderen Beschreibung des Aufbaus der Arbeits-plätze, sind in der Arbeit unberücksichtigt geblieben, da sie deren Umfang gesprengt hätten. Inzwischen gibt es verschiedene Stellen, die die Eignung von Linux für den Desktop-Einsatz untersucht und diskutiert haben. Die Auswahl und Konfiguration der zur Verfügung gestellten Applikationen beeinflussen zwar stark die Nutzbarkeit der vorgestellten Architektur, konnten an dieser Stelle aus Platzgründen nicht mehr aufgenommen werden. Die Diskussionen um die Nichteignung von Linux aufgrund mangelnden Software-Angebotes sind aus gutem Grund fast komplett verstummt.
Die vorgestellte Client-Architektur basiert auf offenen Standards und erlaubt das Einbeziehen anderer Software-Plattformen. So läßt sich mittels des Citrix-Metaframe-Clients eine Verbindung zu Windows-Servern aufbauen, die sich nahtlos in den vorhandenen Desktop eingliedern läßt. Es gibt mehrere Java-Runtime-Umgebungen, die unter Linux zur Verfügung stehen und damit eine ganze Reihe plattformübergreifender Software, wie z.B. Frontends zum SAP-R/3-System, unterstützen. Mit der Fundierung des neuen Betriebssystems der Apple-Macintosh-Architektur[1] auf einer Unix-Basis kann weiterhin eine Anbindung der beschriebenen Clients in dieser Richtung denkbar werden. Im Extremfall tritt vielleicht der "eigene" Desktop der vorgestellten Architektur komplett in den Hintergrund und man benutzt die Thin-Clients als preiswerte, lizenzkostenfreie, einfach skalierbare Lösung zur Anbindung an andere kommerzielle Plattformen, wie Kiosk- oder Point-of-Sale-Produkte. Die in dieser Arbeit genannten Grundlagen für Net-PC's auf Linuxbasis lassen sich darüberhinaus auf die Anwendung für Embedded Systems, welche in Zukunft sicherlich in vielen Geräten des Haushalts,

[1] Mac-OS X basiert auf einem BSD-Kernel

der betrieblichen Produktion und in Überwachungssystemen enthalten sein werden, übertragen.
Die "serverlastige" Auslegung der Architektur - nur eine minimale Bootsoftware wird auf den Thin-Clients installiert - erlaubt ein flexibles Umschalten zwischen unterschiedlichen Betriebsmodi der Endgeräte. Dies gestattet einen einfachen Betrieb der Maschinen mit verschiedenen Betriebssystemen, beispeilsweise einem auf der Festplatte installierten Windows und einer Linuxumgebung über das lokale Netzwerk. Solche Umgebungen können den Betrieb von Schulungsräumen erleichtern, indem eine bequeme Administrationsumgebung zum einen und eine Verbreiterung der Angebotspalette zum anderen erreicht werden können. Vorstellbar wäre auch, normale Bürorechner nachts zu einem Linux-Cluster für paralleles Rechnen zusammenschalten, ohne dafür an den Maschinen selbst wesentliche Änderungen vornehmen zu müssen.
Die Kombination aus der Bootsoftware Etherboot und der freien DHCP-Implementierung des ISC erlauben einen Client-Server-Betrieb parallel zu bestehenden Bootimplementationen. So werden bestehende Infrastrukturen nicht in ihrem Betrieb behindert, so dass zusätzlicher Aufwand durch parallelen Netzbetrieb entfallen kann. Anderseitig kann vermieden werden, dass die Clients von anderen als den vorgesehenen Servern booten können. Dies senkt die Migrationshürde zum Umsetzen eines linuxbasierten Arbeitsplatzumfeldes in der beschriebenen Konfiguration.
Nicht näher eingegangen wurde auf die Vereinfachung der Backups. Durch die zentrale Lösung kann darauf verzichtet werden, von jedem einzelnen Arbeitsplatzsystem ein regelmäßiges Backup der Systeminstallation zu erzeugen. Durch die Gleichartigkeit des Basisfilesystems der Diskless-Clients reduziert sich hier der Aufwand auf das Serverfilesystem inklusive des separaten Bereiches für das Rootfilesystem der Clients. Hierdurch verringert sich der Umfang der zu sichernden Daten erheblich und eine aufwändige Verwaltung der einzelnen Sicherungskopien entfällt durch deren nun weitaus geringere Stückzahl. Darüberhinaus wird das Netzwerk von Sicherungsdatenströmen entlastet und es kann, wie schon eingangs erwähnt, auf dezentrale Sicherungshardware verzichtet werden. Ähnlich stellt sich die Situation bei den festplattenbasierten Systemen dar: Da diese ihr Filesystem von einem Master-Server ableiten und die Konfiguration aus der Datenbank beziehen, sind nur noch die Dateien zu sichern, die bei Updates nicht mit dem Server abgeglichen werden.
Die Administrationstools, welche im Rahmen dieser Arbeit programmiert wurden, bieten einen Einblick in die Implementierung solcher Anwendungen. Die Anbindung an die Datenbank läßt sich aufgrund der unkompliziert zu benutzenden SQL-Schnittstelle ohne Probleme umsetzen. Damit gliedern

sich solche Tools in die Reihe der üblichen Unix-Administrationsskripten ein. Die hier vorgestellten Programme sind aufgabenspezifisch und werden in anderen Einsatzfeldern entsprechend anders aussehen.
Vorstellbar wären z.B. Exportfilter, welche die Einträge in der Datenbank tagesaktuell in Excel-Tabellen umsetzen, damit sie von Buchhaltungen, Entscheidungsträgern oder Kontrollabteilungen innerhalb ihrer Standardanwendungen ausgewertet werden können. Ein anderer Weg liegt in Verknüpfung über Betriebssystemgrenzen hinweg mittels ODBC-Schnittstelle. Übertragungen in SAP-Datenbanken zur Inventarverwaltung und wertmäßigen Erfassung bzw. Schnittstellen zu diesen wären möglich. Unter diesem Blickwinkel sind auch die eingesetzten Tabellenfelder und ihre Aufgliederung zu sehen: Hier möchte man eventuell weitere Daten zu einzelnen Rechnern hinterlegen, wozu z.B. Standortkoordinaten gehören könnten, die das Auffinden einzelner Arbeitsplätze in weitverzweigten, dezentralen Netzwerken erleichtern. Dieses betrifft auch andere Tabellen¸ d.h. einzelne Angaben sind ausführlicher oder weniger umfangreich gewünscht, als in der Beispielumsetzung angegeben. In jedem Netzwerk spiegelt sich immer eine bestimmte Philosophie wider, der Rechnung getragen wurde.

Die Datenbank bzw. deren Steuerungs-Tools lassen sich mit weiteren Fähigkeiten ausstatten, die beispielsweise bereits bei der Zusammenstellung von Rechnern auf Konsistenzprobleme achten können. Auf diese Weise wäre es denkbar, davor zu warnen, inkompatible Komponenten zu kombinieren. Ungünstige Zusammenstellungen, beispielsweise bei der Verwendung von Monitoren und Grafikkarten, lassen sich schon bei ihrer Eintragung anmerken. So kann bereits im Vorfeld einer Anschaffung kontrolliert werden, ob überhaupt noch Platz für bestimmte Komponenten wie Speichermodule und Erweiterungskarten in einem gegebenen System vorhanden ist oder die notwendigen Schnittstellen zur Verfügung stehen.
Die Einführung einer Sicherheitsarchitektur durch auf verschiedenen Ebenen ablaufende Datenverschlüsselung kann durch den Einsatz der Datenbank zur Verteilung öffentlicher Schlüssel erleichtert werden, unabhängig davon, ob sie über DNS verbreitet oder auf die Einzelmaschine kopiert werden sollen.
Zukünftige Label zur Kennzeichnung aller wichtigen Komponenten könnten sich an den neuen Techniken orientieren, wie sie für den Einzelhandel geplant sind. Flexible, kleine Chips, welche auf das Produkt aufgeklebt werden, sind über spezielle Induktionsverfahren ansprechbar, lesbar und zum Teil beschreibbar. Das würde die Erfassung gerade von Bewegungsvorgängen weitestgehend automatisieren.
Im Bereich der Systemüberwachung gibt es eine ganze Reihe von Programmen, die unterschiedliche Schwerpunkte legen. Viele basieren dabei auf dem

Simple Network Management Protocol (SNMP), welches sich problemlos auf der vorgestellten Architektur umsetzen läßt. Die Verwendung des relativ einfachen und recht alten Programms TKined stellt deshalb ein Beispiel aus dem Umfeld der Open-Source-Software dar. Tools zur Netzwerküberwachung wie NetSaint zeigen, welche Möglichkeiten eine offene Architektur bietet, auch eigene Realisierungen von Programmmodulen einzubringen. Gerade hier findet man die Stärken dieses Konzeptes zur Umsetzung einsatzspezifischer Projekte.
Erweiterungen über die bereits definierten Datenfelder hinaus sind aufgrund der freien Lizensierung der SNMP-Implementierung kein Problem. Die Generierung von Konfigurationsdateien, oder so vorhanden, die Anbindung der internen Datenbank der Systemüberwachungsprogramme, erfordern jeweils angepaßte Skripten, wobei auf die Prototypen zurückgegriffen werden kann. Weiterhin ist es vorstellbar, andere zusätzliche Überwachungsdienste auf den Einzelstationen zu installieren, welche andere über SNMP hinausgehende Aufgaben übernehmen können. Die Anlage des Konzepts erleichtert ein solches Vorgehen durch die Zentralisierung der Konfiguration. Die Verknüpfung der Überwachungsinformationen mit den Einträgen aus der Datenbank erlaubt aktuelle Situationsanalysen: Es läßt sich vermeiden, alle beteiligten Netzwerkkomponenten automatisch oder manuell "suchen" bzw. indizieren zu müssen, wodurch z.B. ausgeschaltete oder defekte, jedoch wichtige Systeme herausfallen könnten. Es lassen sich auf diese Weise Wartungs- und Abschaltzeiten definieren und berücksichtigen, die sonst zu verwirrenden oder falschen Warnmeldungen führen könnten.
Nur zum Teil berücksichtigt wurden Erwägungen zur optimalen Verteilung der notwendigen Dienste auf den einzelnen Servern. Zu Beginn des Projektes spielten die Kosten für den Ausbau des Arbeitsspeichers eine entscheidende Rolle, so dass die eingesetzten Boot-Server mit einer Reihe von Aufgaben, angefangen vom DHCP- und NFS-Server bis hin zur Loginmaschine für X-Terminalnutzer, betraut wurden. Da diese Überlegungen aufgrund der aktuellen Entwicklungen am Hardware- und insbesondere am Speichermarkt nicht mehr die zentrale Rolle spielen, empfiehlt sich zur Erhöhung der Gesamtstabilität der Anlage eine Aufteilung der Maschinen in reine Boot- und NFS-Server sowie Loginmaschinen für den Nutzerbetrieb. So kumuliert nicht eine hohe NFS-Belastung aus einer hohen Nutzerlast auf einer Maschine zu Zeiten starker Beanspruchung. Die Erhöhung der verfügbaren Anschlußbandbreiten der Servermaschinen auf ein Gigabit deutet einen weiteren Entlastungspfad an.
Der gezeigte Ansatz liefert nicht ein in sich geschlossenes System, das sich in den im Anhang dieser Arbeit gezeigten Anwendungen erschöpft. Die datenbankgestützte Administration größerer Rechnerpools läßt viele Erwei-

terungen und eine evolutionäre Entwicklung des Systems auf Hard- und Software-Seite zu. Die Offenheit des Ganzen legt dabei den Pfad der Veränderungen nicht fest, sondern läßt weitestgehende Anpassungen an sich verändernde Aufgabenstellungen zu.

Kapitel 21

Zusammenfassung und Danksagungen

Dieser Projektbericht basiert auf den inzwischen sechsjährigen Erfahrungen im Umgang mit dem Betriebssystem Linux sowie den verschiedenen Software-Produkten und Protokollen des Netzwerkbetriebes. Viele Bedingungen haben sich im Laufe der Zeit geändert, die Zahl der betriebenen Rechner ist stark gestiegen, die zur Verfügung stehenden Bandbreiten der zugrunde liegenden Infrastruktur im gegebenen Umfeld haben sich erhöht oder wurden im Zuge der Ausweitung des Betriebes gesteigert. Die Software unterlag zügigen Entwicklungen und Verbesserungen, ihr Funktionsumfang stieg und die Möglichkeiten ihrer Konfigurierbarkeit erhöhten sich. Viele neue Erfordernisse, die sich aus dem Betrieb ergaben, konnten so bereits mit einer aktuelleren Version der jeweiligen Programme abgedeckt werden. Open-Source-Software besitzt neben dem guten Ruf die notwendige Dynamik, um im geschilderten Umfeld eingesetzt zu werden. Viele wichtige Details werden schneller umgesetzt, als es von klassischen Produkten erwartet werden kann. Inzwischen beginnt sich Linux im Bereich der universitären Ausbildung und Forschung fest zu etablieren und stellt eine ernstzunehmende Konkurrenz zu kommerziellen Betriebssystemen dar. Es eignet sich hervoragend als Experimentierplattform und spart erhebliche Betriebsmittel ein.

Begonnen wurde das Projekt mit einer Zahl von sechs Rechnern, die mit Hilfe von Sponsoren beschafft werden konnten. Die unkomplizierte, offene und ergebnisorientierte Arbeitsweise der im Studentenwerk Göttingen insbesondere unter dem Leiter der Abteilung für Organisation Jürgen Dietz gegründeten Internet-AG gestattete das Entwickeln von Lösungen jenseits der etablierten Software-Anbieter. Die Angliederung eines allgemeinen Internet-

zuganges für alle Studierenden der Universität Göttingen an ein Serviceunternehmen studentischer Belange schuf die notwendige Unabhängigkeit, die die Evolution neuer Konzepte zur Erbringung einer inzwischen allgemein anerkannten Dienstleistung im Rahmen von Forschung und Lehre auch nach der späteren Übernahme seitens Universität und Rechenzentrum GwDG sicherstellte. Die wohlwollende Gewährung weitgehender Freiheiten durch die damalige Vizepräsidentin der Universität Frau Prof. Carola Lipp und den Leiter der Gesellschaft für wissenschaftliche Datenverarbeitung Göttingen (GwDG) Herrn Prof. Schneider ermöglichte, dass dieses im Umfang ungewöhnliche Projekt fortgesetzt werden konnte.
Inzwischen konnte das gesammelte Wissen, das im Umgang mit großen Rechnernetzwerken und deren Benutzer gewonnen wurde, auch auf andere Bereiche der Universität übertragen werden. So setzt die mathematische Fakultät das Prinzip der X-Terminals im Bereich der reinen und angewandten Mathematik für die Arbeitsplätze der Wissenschaftler und Studierenden ein. Ein neuer Übungsraum mit 20 Arbeitsplätzen wird auf Basis der DXS-Plattform eingerichtet. Die Staats- und Universitätsbibliothek (SUB) verwendet die Grundlagen dieser Arbeit zum Aufbau und der Erweiterung ihres grafischen OPAC-Systems[1]. In zwei Bereichen, einem gemeinsam mit dem Rechenzentrum genutzten Kursraum von 16 Arbeitsplätzen und einem neu eingerichteten CIP-Pool[2] der juristischen Fakultät wird eine Dual-Bootlösung betrieben, welche die parallele Nutzung von Windows-NT (bzw. Windows 2000) und Linux auf einer Maschine erlaubt. Auf diese Art und Weise konnte sich das vorgestellte Konzept inzwischen im Arbeitsbereich der Internet-Hotline der Universität an derzeit über 400 Arbeitsplätzen und 10 Servern und den genannten Bereichen der Fakultäten und der SUB bewähren. Damit hat das Projekt den Status einer exotischen Sonderlösung verlassen und ein weites Anwendungsspektrum erlangt.
Neben den Erfahrungen des Netzwerkbetriebes und seiner Datenbanksteuerung, die im Mittelpunkt dieser Arbeit standen, wurde auch etliches Wissen hinsichtlich der Konfiguration der Anwendungssoftware, der Gestaltung der grafischen Desktops und der Wünsche nach bestimmten Features gesammelt. In diesem Zuge entstanden einige nützliche Tools zur Benutzerverwaltung, Steuerung und Personalisierung von Accounts und der Anpassung des Linuxdesktops an den Massenbetrieb. Diese Werkzeuge runden den Einsatz des vorgestellten Konzeptes ab und erlaubten den Ausbau auf die nun erreichte Größe. Die inzwischen recht weite Verbreitung im privaten Einsatz und in Unternehmen sowie die starke öffentliche Wahrnehmung von Linux erlaubt nun auch einen verstärkten Einsatz im Bereich der Forschung, Aus-

[1] Benutzerschnittstelle zum Zugriff auf die Kataloge und Ressourcen der Bibliothek
[2] Computer Investitions Programm des Landes

bildung und öffentlichen Verwaltung, wozu das vorgestellte Konzept einen Beitrag leisten will.
Insbesondere möchte ich dem Förderer des Einsatzes von Linux im Bereich der GwDG und seine vielfältige Unterstützung, dem Herrn Eberhard Mönkeberg danken. Mein Dank gilt ebenso dem Herrn Prof. Schneider für die Möglichkeit der Umsetzung des Projektes. Hervorheben möchte ich an dieser Stelle auch die Unterstützung seitens der Mitarbeiter der Internet-Hotline ohne die eine Aufgabe dieser Größenordnung nicht umsetzbar gewesen wäre. Danken möchte ich auch den vielen Korrekturlesern dieser Arbeit, allen voran Sebastian Sigel.

Anhang A

CBROM und AMIBCP

A.1 Vorbemerkungen

BIOS-Utilities Die beiden Programme **cbrom.exe** und **amibcp.exe** sind DOS-Utilities, die zur Modifikation von PC-BIOSes verwendet werden können. Überlicherweise werden diese von den Mainboardherstellern verwendet, um bestimmte Zusatzkomponenten, wie zusätzliche IDE- oder SCSI-Kontroller, Virussoftware oder VGA-Treiber dem Core-BIOS hinzuzufügen. Das Bootlogo des Herstellers läßt sich auf diese Weise entladen und evtl. modifiziert wieder einbinden.
Möchte man Veränderungen am BIOS vornehmen, ist eine gewisse Vorsicht angebracht, da bei falschem Inhalt des Bausteins ein Totalausfall des Mainboards die Folge sein kann. Geht man jedoch umsichtig vor und legt sich evtl. den Flash-Baustein mit dem Original-BIOS sicher beiseite, kann im Fall des Fehlschlages auf das Backup zurückgegriffen werden.

Backup des Flash-Bausteins Eine etwas unorthodoxe Version ein Backup zu erzeugen kann wie nun beschrieben erfolgen: Man startet den Rechner, der mindestens über eine bootfähige Festplatte oder Diskettenlaufwerk verfügt und sichert den Inhalt des BIOS mit den mitgelieferten Flash-Utilities[1] in eine Datei. Oder man kopiert sich das aktuelle BIOS von der Website des Mainboardherstellers. Dann wechselt man vorsichtig im laufenden Betrieb den Flash-Baustein aus[2] und steckt ein identisches Modul an seine Stelle. Nun startet man das Flash-Utility erneut oder wechselt aus dem laufen-

[1] **awdflash.exe** für AWARD-BIOSes und **amiflash.exe** oder **flash830.exe** für AMI-BIOSes

[2] Meist ist es sinnvoll ihn im ausgeschalteten Betrieb soweit zu lockern, dass die Anschlüsse gerade noch Kontakt zum Sockel haben.

den Tool in die Schreiboption. Dann führt man den Schreibprozess durch und flasht das BIOS-File in den neu eingebauten Baustein. Das ausgebaute Flash-Memory legt man nun sicher beiseite und hat im Fall eines Misserfolges beim Patchen des BIOS eine Rückgriffsmöglichkeit. Man sollte jedoch bei dieser Art von Experimenten bedenken, dass ein Garantieverlust für das Mainboard eintreten kann.

Generelles Vorgehen Auch für alle weiteren nun beschriebenen Vorgänge benötigt man das Image des BIOS als Datei, da weder **cbrom.exe** noch **amibcp.exe** direkt auf dem Flash-Baustein operieren.
Achtung: Man sollte Etherboot mit den Optionen `-DASK_BOOT` oder `-D EMERGENCYDISKBOOT`[3] kompilieren, da man sonst nicht mehr auf Diskette oder Festplatte zugreifen kann. Der Code, den man dem BIOS hinzufügt, wird noch vor dem ersten Platten- oder Floppyzugriff ausgeführt, außer es steht eine spezielle BIOS-Option zur Verfügung. Sind PCI-Netzwerkkarten im Einsatz, kann das Booten von Festplatte oder Diskettenlaufwerk durch das Entfernen der Karte wieder ermöglicht werden. Die aktuelle Version von Etherboot erlaubt durch eine modifizierte Interrupt-Architekur eine verbesserte Zusammenarbeit mit dem System-BIOS, so dass bereits durch geeignete Einstellungen der Reihenfolge der bootfähigen Geräte die Ausführung von Etherboot unterbunden werden kann.
Obwohl in einigen Konfigurationen kein (E)EPROM auf der Netzwerkkarte installiert wird, muss in vielen Fällen ein EPROM auf der Karte mit dem Diagnose- oder Setuptool eingeschaltet werden. Dabei spielt die eingeschaltete Größe meistens eine untergeordnete Rolle. Sonst kann es passieren, dass die Bootsoftware die Netzwerkkarte nicht erkennt.
Weiterhin sollte beachtet werden, dass Etherboot bei PCI-Netzwerkkarten nur ausgeführt wird, wenn vom Code die richtige PCI- und Vendor-ID der Netzwerkkarte erkannt werden. Damit ist es dann aber denkbar, ausreichenden Platz im BIOS-Flash vorausgesetzt, mehrere Etherboottreiber gleichzeitig unterzubringen. Damit wäre das Wechseln der Kartentypen, das Vorhandensein des entsprechenden Treibers vorausgesetzt, mit wenig Problemen verbunden.
Nach der Modifikation des BIOS mittels der beiden nachstehend beschriebenen Tools muss die BIOS-Datei wieder in den Baustein geflasht werden. Manchmal wird es notwendig sein, nach dem Neustart des Rechners die BIOS-Einstellungen erneut vorzunehmen. Bei einigen Boards ist es außerdem notwendig die Bootreihenfolge so einzustellen, dass als erstes die LAN-Option ausgeführt wird. Bei den meisten Boards spielt die Bootreihenfolge

[3] Nähere Erklärungen zur Konfiguration der Compileparameter von Etherboot sind im Abschnitt (B.2) zu Etherboot zusammengefaßt.

für das Starten von Etherboot jedoch keine Rolle, so dass o.g. Vorsichtsmassnahmen beherzigt werden sollten.

A.2 AWARD-BIOS

cbrom.exe arbeitet kommandozeilenorientiert. Der Aufruf des Programms ohne Parameter zeigt alle zur Verfügung stehenden Kommandozeilenoptionen an. Mit "cbrom bios.bin /d" wird ermittelt, wieviel Platz noch im ROM-File vorhanden ist und für unseren Code zur Verfügung steht. Das BIOS im Flash-Rom liegt komprimiert vor und **cbrom** komprimiert den generierten Code, wenn es diesen dem BIOS hinzufügt. Man benötigt deshalb, in Abhängigkeit vom Treiber der eingesetzten Netzwerkkarte, 8 bis 20 kByte freien Speicherplatz. Ist nicht genügend vorhanden, können nicht benötigte BIOS-Komponenten entfernt werden: Das Logo des Herstellers oder der Symbios/NCR SCSI-Code werden für festplattenlose Systeme nicht benötigt[4] . "cbrom bios.bin /[pci|ncr|logo|isa] release" entfernt solche nicht benötigten Teile.
Möchte man bestimmte BIOS-Komponenten, die man entfernt zur Sicherheit separat speichern, kann dieses mit "cbrom bios.bin /[pci|ncr|logo|isa] extract" geschehen. Die BIOS-Komponente wird dann unter ihrem ursprünglichen Dateinamen gesichert.
Mit dem Kommando "cbrom bios.bin /[pci|isa] bootimg.rom [D000:0]" fügt man den kompilierten Etherbootcode zum BIOS hinzu. Das bootimg.rom ist der Code, den man sonst in ein EPROM brennen würde. Die Option [pci/isa] hängt vom Typ der Netzwerkkarte ab. Wenn man eine ISA-Karte verwendet, gibt man **cbrom** darüberhinaus die Speicheradresse[5] mit, an die der Code wärend des Bootvorgangs kopiert werden soll.

A.3 AMI-BIOS

Das AMI-Tool **amibcp.exe** läßt sich im Gegensatz zu **cbrom.exe** interaktiv menügesteuert bedienen: Man startet das Tool ohne Kommandozeilenparameter und lädt das BIOS-File über den ersten Menu-Punkt: "Load BIOS from Disk File". Bearbeitet wird das BIOS über die dritte Menüoption "Edit BIOS Modules". Am unteren Bildschirmrand wird die noch verfügbare Speichermenge für Erweiterungen angeziegt. Sollte dieser nicht ausreichen,

[4] Das SCSI-BIOS stellt Bootfunktionalität für BIOS-lose SCSI-Controller her und wird im Linuxbetrieb nicht benötigt, da der Kernel alle IO-Aufgaben übernimmt.

[5] Im Bespiel mit "D000:0" angegeben, überlicherweise im Bereich von C800:0 bis DC00:0

kann man versuchen überflüssige BIOS-Komponenten (Hersteller-Logo, Virenschutzsoftware ...) zu entfernen.
Mit der "INS"-Taste kann man Zusatzmodule einfügen. Das Modul kann "compressed" eingefügt werden, was bevorzugt werden sollte. Mit der "ESC"-Taste verläßt man den Edit-Bereich und mit Menu-Punkt Zwei: "Save BIOS File to Disk" speichert man das veränderte BIOS wieder in eine Datei.

Anhang B

Erstellen von Bootimages

B.1 Überblick zur einsetzbaren Software

Im Wesentlichen gibt es zwei verfügbare, freie Pakete im Sinne der Open-Source-Lizenzen zum Erstellen von Bootimages. Diese Startprogramme müssen möglichst klein sein, damit sie auf einem EPROM Platz finden oder dem Mainboard-BIOS hinzugefügt werden können. Eines dieser Pakete ist Etherboot. Eine ganze Reihe von Entwicklern aus dem Umfeld der freien Software und einigen interessierten Firmen sorgen für dessen beständige Aktualität. Zur Entstehungszeit dieser Arbeit war die Version 5.0.X[1] aktuell, so dass sich alle hier gemachten Bemerkungen auf diese Version beziehen werden. Im ersten Abschnitt zum Thema werden ausführlich die verschiedenen Compile-Time-Options von Etherboot besprochen. Im folgenden Abschnitt wird darauf eingegangen, wie man ein ROM-Image erstellen kann und bestimmte Anpassungen für PXE und "Vendor-Code-Identifier" vornehmen kann.
Eine Alternative zu Etherboot liegt mit dem Netboot-Paket vor, welches auf die Netzwerktreiber der jeweiligen Hersteller aufsetzt und etwas anders als Etherboot funktioniert. Jedoch wird es mit der breiten Unterstützung fast aller Netzwerkkarten im Etherbootpaket nicht mehr weiterentwickelt. Eine kurze Anleitung zu seiner Benutzung folgt nach Etherboot. Zum Ende des Kapitels wird auf das Programm zum "Taggen" des Kernels eingegangen, welches von beiden Paketen verwendet wird, da sie auf gemeinsamer konzeptioneller Grundlage aufbauen.

[1] Die neuesten stabilen und Developmentversionen können über *http://etherboot.sourceforge.net* bezogen werden und sind auch über jedes größere FTP-Archiv erreichbar.

B.2 Etherboot

B.2.1 Installation des (Source-)Paketes

Viele Linuxdistributionen enthalten bereits das Etherbootpaket im Lieferumfang. Wenn jedoch nicht die evtl. benötigte aktuellste Version installiert ist oder besondere Optionen erforderlich sind, empfiehlt sich das Selbstkompilieren. Das Paket wird "tar -xpzf etherboot.tgz" entpackt. Es gibt ein umfangreiches Verzeichnis mit Anleitungen in verschiedenen Formaten unter *doc*[2]. Das Verzeichnis *src* enthält alle notwendigen Dateien. Hier erhält man nach dem Aufruf von **make** für jeden Typ Netzwerkkarte ein **.rom* Image, wobei unterhalb von *bin32* die Abbilder für 32 bit-Maschinen[3] liegen.

B.2.2 Allgemeine Einstellungen

Alle zentralen Optionen für Etherboot lassen sich in der Datei *Config* im *src*-Verzeichnis des Quellpaketes vornehmen. Alle zur Verfügung stehenden Optionen sind zu Beginn dieser Datei kurz erläutert. Weitere Ausführungen entnimmt man der umfangreichen Dokumentation. Die Aus- bzw. Abwahl einzelner Optionen entscheidet über den Umfang des später erzeugten ROM-Images, welche üblicherweise zwischen 16 und 32 kByte[4] liegen wird.

```
# For prompting ...
CFLAGS32+=     -DMOTD -DIMAGE_MENU
[...]
```

In dieser Datei wird auch festgelegt auf welche Weise das Kernelimage geladen werden soll, ob dieses per TFTP oder mittels NFS geschehen soll. Die NFS-Unterstützung vergrößert den erzeugten Code, gestattet aber auf dem Server den (unsicheren) TFTP-Dienst abzuschalten, womit sich auch die Fehlerwahrscheinlichkeit senken läßt.

```
# Change download protocol to NFS, default is TFTP
CFLAGS32+=      -DDOWNLOAD_PROTO_NFS
```

Möchte man die Meldung zur Auswahl der Bootart (Booten aus dem Netzwerk oder von einem lokalen Device) modifizieren, so geschieht dieses in der Datei *etherboot.h* welche im selben Verzeichnis wie die Konfigurationsdatei

[2] Wegen ihres Umfanges muss die Dokumentation separat von der Etherboot-Homepage geladen werden

[3] Unter *bin16* liegen die Images für 16 bit Maschinen, 286er werden jedoch kaum noch eingesetzt.

[4] EPROM's müssen also die Größe von 128 kbit oder 256 kbit aufweisen, bzw. das BIOS-Flash-Rom über entsprechend freien Platz verfügen

liegt. Möchte man zusätzlich den String ändern, mit welchem sich Etherboot per "Vendor-Code-Identifier" meldet, bzw. an welchem Etherboot gültige DHCP-Antworten unterscheidet, ist die Datei *main.c* anzupassen. Zum Wirksamwerden letztbeschriebener Optionen ist jedoch das Einschalten in der zentralen Konfigurationsdatei Bedingung.

```
# For prompting and default on timeout
CFLAGS32+=     -DASK_BOOT=3 -DANS_DEFAULT=ANS_NETWORK
[...]
# Enabling this makes the boot ROM require a Vendor Class Iden-
# tifier of "Etherboot" in the Vendor Encapsulated Options
CFLAGS32+=      -DREQUIRE_VCI_ETHERBOOT
```

B.2.3 Kompilation

Zur Erzeugung aller **.rom*-Images genügt der einfache Aufruf von **make** im Sourceverzeichnis. Diese Images stehen anschließend im Unterverzeichnis *bin32* zur Verfügung. Zum Testen der generierten Boot-ROM-Abbilder empfiehlt es sich, diese zuerst mit einer Bootdiskette zu testen. Einzelne Diskettendateien werden direkt durch den Aufruf von **make bin32/name_der_netzwerkkarte.fd0** erzeugt und auf die im Laufwerk befindliche Diskette geschrieben. Der Unterschied zum später eingesetzten ROM-Image liegt im vorgeschalteten Disketten-Bootheader.
Das Erzeugen von PXE-Images, welche durch die auf Clients evtl. bereits vorhandene Bootsoftware geladen werden, geschieht auf analoge Weise. **make bin32/name_der_netzwerkkarte.pxe** erzeugt eine entsprechende Datei. Dieses kann direkt in Zusammenarbeit mit PXE getestet werden, da eine Festinstallation auf dem Client unnötig ist.

B.2.4 Multiboot-Anpassungen

Interessant sind die einbaubaren Anpassungen von Etherboot an Multibootumgebungen, die neben dem klassischen Start aus dem Netzwerk auch Optionen zum Booten von Festplatte, Diskette oder CD-ROM-Laufwerk zur Verfügung stellen können. Dieses kann zum einen direkt vor dem Start aus dem Netzwerk geschehen. Hier wird die Unabhängigkeit von der Verfügbarkeit eines Bootservers sichergestellt. Die Flexibilität späterer Anpassungen ist jedoch begrenzt, da dann der Bootcode eines jeden Clients ausgetauscht werden muss. Begibt man sich in die Abhängigkeit der Verfügbarkeit der Bootserver läßt sich über diese eine ausgefeilte Bootmenu-Struktur verwirklichen und bei Bedarf dynamisch über den DHCP- oder BOOTP-Server anpassen. Diese Veränderungen ziehen dann kein Updatebedarf auf Clientseite

nach sich. Die Kombination beider Varianten macht keine Probleme, könnte aber am Ende zur Verwirrung des Anwenders führen, da dann mehrfache Auswahlen präsentiert werden.

B.3 Netboot

Die letzte verfügbare Version des Netbootpaketes trägt die Nummer 0.9[5]. Gegenüber der Vorgänger-Version 0.8 (welche aufgrund ihres einfacheren Designs kleinere Bootimages erzeugt) wird in der aktuellen Fassung auch die Komprimierung großer (Packet/NDIS-) Treiber unterstützt. Die maximal unterstützte EPROM-Größe liegt zur Zeit bei 64 kByte (27C512[6]).
Dieses Paket liegt als komprimiertes TAR-Archiv vor, muss entpackt, danach mit dem Kommando **configure** angepaßt und anschließend mit **make** kompiliert werden. Ein ROM-Image (hiermit wird automatisch auch ein Floppyimage zu Testzwecken erzeugt) erstellt man durch **make bootrom**. Wenn die benutzte Netzwerkkarte nicht in der umfänglichen Liste (zur Zeit 129 Einträge), sollte der Pfad zum Paket- oder NDIS-Treiber der Netzwerkkartendiskette (und die Vendor und PCI-ID[7]) bekannt sein. Anschließend kann das erzeugte Abbild mit der Endung *.flo* auf eine Diskette "dd if=image.flo of=/dev/fd0" kopiert und auf dem Thin-Client getestet werden.

B.4 "mknbi(-linux)"

Es stehen inzwischen zwei Versionen zur Verfügung, ein C-Programm, welches kompiliert werden muss und als neuere Entwicklung ein Perl-Skript. Beide Tools erlauben neben der Erstellung von Bootimages für Linux auch die Unterstützung weiterer Betriebssysteme, z.B. DOS. Die Manualpage kann mit "man mknbi" eingesehen werden.
Dem Netbootpaket liegen die Sourcen für ein **mknbi**[8] bei, welche wiederum durch **make** erzeugt werden können. Das Kerneltagging geschieht mit: **mknbi -k zImage -o bootimg -d /nfsroot/dxt -i rom**. Die Optionen geben nacheinander das Kernelfile, das Outputfile (der dann netzbootfähi-

[5] Netboot und Dokumentation kann über http://netboot.sourceforge.net bezogen werden

[6] Die Größe von (E)EPROMS wird meistens in Bit angegeben, so dass ein 32 kByte ROM-Baustein mit 256 kBit bezeichnet wird, was sich auch im Aufdruck bis 512 kBit direkt widerspiegelt. Siehe hierzu auch 9.3.2

[7] Sie werden beim Start des Rechners am Ende des Bootvorganges in einer Liste angezeigt bevor das OS geladen wird.

[8] make netboot image

ge Kernel) und das NFS-Root an. Die letzte Option übergibt dem Kernel die Information, dass er seine IP-Konfiguration über die DHCP/BOOTP-Anfrage des Boot-ROM's bezieht.
Das aktuellere Tool kann als separates Paket von der Etherboot-Homepage[9] bezogen werden.

Schalter	**Erklärung seiner Funktionalität**
$--ip = option$	Bestimmt auf welche Weise der Kernel die IP-Konfiguration erhält. In den meisten Fällen wird man "rom" spezifizieren, so dass die von Etherboot bezogenen Daten an den Kernel per Kommandozeile weitergereicht werden
$--outputfile = file$	Dateiname des netzbootfähigen Kernels, wenn nicht die Ausgabeumleitung der Shell benutzt werden soll
$--param = string$	Erlaubt die Ersetzung des Default-Parameterstrings, welche die nachfolgenden Optionen überschreibt
$--append = string$	Hängt den angegebenen String an den vorliegenden Parameterstring an
$--rootdir = path$	Gibt den Namen des Verzeichnisses an, welches mittels NFS nach dem Start des Kernels gemountet werden soll
$--first16$	Verwenden des alten 16 bit First Stage Setup Programms, welches bei Problemen testweise verwendet werden kann

Tabelle B.1: Liste der Kommandozeilenschalter von **mknbi-linux**

Im Verzeichnis *mknbi* liegt der Code für das Tool zum Taggen[10] des Kernels. Dieses wird durch **make install** in */usr/local/bin* installiert. Dieses Tool ist perlbasiert und damit weitgehend plattformunabhängig. Mit **mknbi-linux –outputfile /nfsroot/xtbootimg** wird das Kernelimage "bzImage" mit einem entsprechenden Header versehen.
Dem Netzwerkkernel werden eine ganze Reihe von Komfigurationsparametern per Kernel-Command-Line mitgeteilt, wozu die eben beschriebene IP-Konfiguration und Angabe des Root-Verzeichnisses zählen. Sollen darüber-

[9] *http://etherboot.sourceforge.net*

[10] Das Kernelimage muss mit einem speziellen Bootheader versehen werden, damit es nach dem Übertragen mittels TFTP oder NFS gestartet werden kann. Dieses Tagging ist softwarespezifisch und funktioniert nur im Zusammenhang mit Etherboot bzw. Netboot.
Wird eine andere BOOTP/TFTP-Software, beispielsweise von 3COM verwendet, sollten die Tools des entsprechenden Herstellers eingesetzt werden.

hinaus Daten zur Einstellung bestimmter Parameter, z.B. für das Framebufferdevice und andere Geräte übermittelt werden, sollte die Zahl der erlaubten Parameter und die Länge des Parameterstrings[11] erhöht werden.

[11] Dieses geschieht in den Dateien der Kernelsourcen *linux/main.c* (boot command-line arguments) und *arch/i386/kernel/setup.c* (#define COMMAND_LINE_SIZE).

Anhang C

Der DHCP-Server

C.1 Funktion

DHCP ist ein UDP-basiertes IP-Netzwerkprotokoll, über das grundsätzliche Daten zur Konfiguration eines Clientsystems übertragen werden können. Es stellt eine Erweiterung und Ergänzung des BOOT-Protokolls dar, welches wichtige zusätzliche Funktionen implementiert, die im Zuge dieses Projektes ausgenutzt wurden. Neben den klassischen Parametern, wie Hostname, IP-Adresse, Netzmaske und Gateway, zählt dazu die Verwendung einer Reihe von weiteren IP-Variablen: X-Display- , Time-, Swap-, NIS-Server und die Unterstützung von Vendor-Code-Identifiern. Darüberhinaus können in einem Stringfeld z.B. die Konfiguration von Grafikkarte und Monitor incl. Mausanschluß übermittelt werden. Ein weiteres Feld nimmt Kommandozeilen auf, die an die Datei *boot.local* angehängt und ausgeführt werden, auf. So lassen sich die Audiokomponenten konfigurieren oder zusätzliche Gerätetreiber laden.

C.2 Vorläufer: Der "bootpd"

Inzwischen findet der **bootpd** kaum noch Verwendung, da er vom DHCP abgelöst wurde. In einigen Fällen möchte man vielleicht trotdem auf ihn zurückgreifen, weshalb seine Konfiguration im folgenden kurz beschrieben wird. Die Parameter des **bootpd** lassen sich am besten anhand der */etc/bootptab*, der zentralen Konfigurationsdatei des **bootpd** erläutern:

```
.default:\
        :td=/nfsroot/:\
        :hd=/nfsroot/:\
```

```
        :rp=/nfsroot/dxt:\
        :bf=xtbootimg:\
        :ht=ether:\
        :sm=255.255.0.0:\
        :gw=172.16.98.254:\
        :ns=172.16.98.1:\
        :dn=your.domain:\
        :vm=rfc1084:\
        :hn:to=-18000:

terminal01:tc=.default:\
        :ha=0020AF5733BB:\
        :ip=172.16.189.111:
```

Die einzelnen Optionen werden in folgender Tabelle erklärt. Wenn für alle Terminals die gleichen Parameter gelten, könnten diese in einer gemeinsamen Konfiguration definiert werden (hier .default). Es dürfen auch mehrere solcher Konfigurationen angegeben und mit dem 'tc'-Feld entsprechend zugewiesen werden.

Kürzel	Typ	Erklärung
hd	string	"Homedirectory" des Bootfiles
td	string	TFTP-Rootverzeichnis
bf	string	Boot-File (Netkernel)
ht	string	Hardware Type
sm	ip	Netmask
gw	ip	Gateway
ns	ip	Nameserver
dn	string	Domainname
vm	string	Vendor Magic Number
hn	bool	Hostname übermitteln
ha	binary	Hardware Adresse (MAC)
ip	ip	IP-Nummer des Clients
to	integer	Timeout

Tabelle C.1: Liste der **bootpd** Optionen

Unterscheiden sich die Konfigurationen für die einzelnen Terminals stark, werden hinter dem Namen des Terminals alle Parameter angegeben. Der Zeilenumbruch wird durch den Backslash versteckt, um die Lesbarkeit der Datei für die Administration zu erleichtern. Alle Felder müssen mit Doppelpunkten voneinander getrennt sein. Als weitere Optionen können Time-,

Print-, YP-Server, YP-Domain[1] etc. übertragen werden.
Dabei muss bedacht werden, dass mit dem BOOTP-Paket[2] nur eine bestimmte Menge an Informationen transferiert werden kann. So können nicht alle dargestellten Optionen auf einmal übertragen werden. Dieses läßt sich jedoch über sogenannte BOOTP-Extensions realisieren. An dieser Stelle bietet sich jedoch die Verwendung des DHCP an, welche im nächsten Abschnitt beschrieben wird.
Die Felder für das TFTP[3] weisen schon auf eine weitere Funktion hin, die in der beschriebenen Konfiguration benötigt wird, aber nicht zwingend gemeinsam mit BOOTP verwendet werden muss.

C.3 Konfiguration

C.3.1 Standardeinträge

Die DHCP-Konfigurationsdatei */etc/dhcpd.conf* hängt in ihrem Umfang vom Einsatzziel ab: So werden für Diskless X-Stations vielleicht eine Reihe zusätzlicher Daten, z.B. über Informationen zum Font-, Print-, Swap- und NIS-Server übertragen, die für den X-Terminalbetrieb nicht erforderlich sind. Entsprechende eigene Optionstrings werden eingeführt, um bestimmte Textfelder[4] z.B. zur Konfiguration des X-Servers zu übertragen:

```
option nis-domain   string;
option nis-servers  ip-address [, ip-address... ];
option font-servers ip-address [, ip-address... ];
option lpr-servers  ip-address [, ip-address... ];
option swap-server  ip-address [, ip-address... ];

option nis-domain   string;
option nis-servers  ip-address [, ip-address... ];
option font-servers ip-address [, ip-address... ];
option lpr-servers  ip-address [, ip-address... ];
option swap-server  ip-address [, ip-address... ];
```

[1] "YP" steht für "Yellow Pages" welches ein eingetragenes Warenzeichen der British Telecom ist. Deshalb findet man dieses Kürzel in vielen Kommandos (z.B. **ypbind**). Es wird inzwischen durch NIS (Network Information System) ersetzt.
[2] üblicherweise 572 Byte incl. Header
[3] Trivial FTP, sehr im Funktions- und Sicherheitsumfang reduziertes FTP-Protokoll
[4] Variablentyp "string"

C.3.2 Benutzerdefinierte Optionen

DHCP bietet die Möglichkeit eigene sogenannte Vendoroptionen aufzunehmen, d.h. es können zusätzliche Optionen zu den bereits vorhandenen definiert werden. Hierfür steht der Codenummernbereich von 128 - 255 zur Verfügung. Wenn man umfangreiche Informationen übertragen will, sollte man die Paketgröße des BOOTP-Replypakets über den Standard hinaus erhöhen. Diese Optionen werden zu Beginn der Konfigurationsdatei[5] des **dhcpd** definiert. Die folgende Liste ist eher als Beispiel, denn als vollständige Aufzählung zu verstehen, da für gegebene Aufgaben die notwendigen Felder auch völlig anders aussehen können.

```
# -- much information to be transferred --
dhcp-max-message-size 1024;
# -- user defined options --
option o128 code 128              = string;
option o129 code 129              = string;
option menudflts code 160         = string;
option motdline1 code 184         = string;
option menuline1 code 192         = string;
option menuline2 code 193         = string;
option menuline3 code 194         = string;
option bootlocal-script code 221 = string;
option x-server-defs code 222     = string;
option start-x code 223           = string;
option start-snmp code 224        = string;
option start-sshd code 225        = string;
option start-xdmcp code 226       = string;
option start-cron code 227        = string;
option crontab-entries code 228   = string;
option start-rwhod code 229       = string;
option start-lpd code 230         = string;
option start-dnetc code 231       = string;
```

Wobei folgende Variablen verwendet werden können: *string, integer, boolean, text, ip-number*. Diese können auch zu Arrays zusammengefaßt werden. Die Option 128 definiert ein "Magic-Packet", welches die Auswertung von Menu-Optionen für Etherboot einschaltet. Standardwerte für die Menu-Auswahl, d.h. welches Feld nach einem gewissen Timeout gestartet werden soll, werden mit der Option 160 festgelegt. Die Zusammensetzung des Menüs, welches nach dem Kontakt von Etherboot mit dem DHCP-Server

[5] üblicherweise */etc/dhcpd.conf*

angezeigt wird, geschieht durch die Optionen 192 und folgende. Hierbei wird für jede Menu-Zeile ein eigenes Feld benötigt. Mittels der Option 129 können Parameter zum Kernelstart übermittelt werden, welche z.B. auch den Rootpfad enthalten können[6]. Eine "Message of the day" (Textfeld über der Menu-Auswahl) kann durch das Setzen der Option 184 erfolgen. Stehen nicht genügend Optionsfelder zur Verfügung, können die eben beschrieben Optione auch "wiederverwendet" werden, wenn eine Unterscheidung mittels Vendor-Code-Identifier[7] getroffen wird. Anschließend verwendet man diese Optionen genauso, wie die bereits im DHCP vordefinierten:

```
group {
  option x-server-defs "de PS/2 psaux 65 90 1024x768 vesa 16";
  option bootlocal-script "insmod agpgart";
  option start-x "indirect";
  option start-snmpd "yes";
  [...]
  host dxs01 {
  [...]}
  }
```

Das "group" Statement hilft hierbei, neben der Unterteilung nach Subnetzen, Konfigurationsblöcke mit gleichen Parametern zusammenzufassen. So muss nicht für jeden Host die gesamte Palette wiederholt werden.
Eine Beispielkonfiguration könnte wie folgt aussehen:

```
option domain-name "stud.uni-goettingen.de goe.net gwdg.de";
 filename "/nfsroot/bootimg";
 use-host-decl-names on;
 default-lease-time 72000;
 max-lease-time 144000;
 subnet 134.76.60.0 netmask 255.255.252.0 {
 option domain-name-servers 134.76.60.21, 134.76.60.100;
 option ntp-servers ntps1.gwdg.de, ntps2.gwdg.de;
 option font-servers dionysos.stud.uni-goettingen.de;
 option x-display-manager s4, s5, s6, s9, s10, s11, s12;
 # option netbios-name-servers 134.76.63.252;
 option routers 134.76.63.254;
 option broadcast-address 134.76.63.255;
 # range 134.76.60.201 134.76.60.253;
 }
```

[6] alternativ zum "tagging" mittels **mknbi-linux**
[7] siehe hierzu den Unterabschnitt C.3.3

```
group {
 option x-server-defs "de IMPS/2 psaux 96 100 800x600 nv 16";
 option bootlocal-script "modprobe -qa agpgart esssolo1";
 option start-dnetc "no";
 option start-cron "no";
 host dxs02 {
 hardware ethernet 00:00:1C:D2:87:DF;
 fixed-address 134.76.60.64; }
 [...]
```

C.3.3 Die Verwendung von Vendor-Code-Identifiern

Vendor-Code-Identifier sind als feste Optionen für DHCP definiert: "vendor-class-identifier" für die Identifizierung des Clients durch den Server und "vendor-encapsulated-options" zur Identifizierung des Servers seitens des Clients.
Auf diese Weise lassen sich die DHCP-Anfragen verschiedener Clients voneinander differenzieren, so dass es gelingt an eine Maschine in Abhängigkeit vom anfragenden Client verschiedene Werte für die gleich Option zurückzuliefern. Dieses ist zwingend notwendig, wenn PXE und Etherboot hintereinander verkettet booten sollen, da PXE zwar die identische IP-Konfiguration erhält, aber anstelle des Kernelimages das Etherboot-PXE-Image zur Ausführung laden soll. Dieses sieht man an untenstehendem Beispiel: Es werden bestimmte Optionen nur gesetzt, bzw. die Default-Option überschrieben, wenn in der Anfrage des Clients ein bestimmter String identifiziert werden kann.
Weiterhin kann diese Technologie durch **dhclient** verwendet werden (siehe D.2), wodurch sich die Zahl der benötigten Variablen reduzieren läßt. Das kann notwendig werden, wenn sehr viele verschiedene Optionen übermittelt werden sollen.

C.3.4 Interaktion mit PXE und Etherboot mittels VCI

```
# -- vendor identifier dependend settings --
class "Etherboot" {
 match if substring (option vendor-class-identifier, 0, 9)\
     = "Etherboot";
 option o128 E4:45:74:68:00:00;
 option motdline1 = "Welcome to Guru Labs classroom";
 option vendor-encapsulated-options \
     3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
```

```
class "PXEClient:" {
 match if substring (option vendor-class-identifier, 0, 10) \
     = "PXEClient:";
 filename "/nfsroot/dxs/boot/eb-3c905c.pxe";
}
[...]
```

Anhang D

Verwendete Skripten

D.1 Das "boot"-Skript

Das Skript *(/nfsroot/dxs)/etc/init.d/boot* wird vom Init-Prozess als Erstes aufgerufen. Es orientiert sich am Boot-Skript der SuSE-Distribution, wurde aber an die Anforderungen der Thin-Clients angepaßt. Wichtige Netzwerkparameter und Variablen werden bereits hier konfiguriert, da sie Voraussetzung für das Starten vieler weiterer Dienste bilden. Dies geschieht nach dem Mounten zusätzlicher Filesysteme (nur für DXS notwendig) mittels **dhclient**. Zu diesem Zeitpunkt sendet **dhclient** kein Broadcast, sondern stellt die Anfrage an den Bootserver direkt. So läßt sich eine Fehlkonfiguration des Clients durch evtl. weitere im Netz vorhandene offene DHCP-Server vermeiden und die Strategie der "vendor code identifier" zur Absicherung fortsetzen. Das zugehörige Konfigurationsskript ist im nächsten Abschnitt (D.2.2) abgedruckt.

```
#! /bin/sh
#
# Copyright (c) 1996 SuSE GmbH Nuernberg, Germany.
# All rights reserved.
#
# Author: Florian La Roche <florian@suse.de>, 1996
#         Werner Fink <werner@suse.de>, 1996
#         Burchard Steinbild <bs@suse.de>, 1996
#
# Modifications for use with Diskless X Stations
#
# Copyright (c) 2001, Version: 7.1.3a
#
# Author: Dirk von Suchodoletz <dirk@goe.net>, 12-12-2001
#
#
```

```
# /etc/init.d/boot
#
# first script to be executed from init on system startup
#
# At first we create a ramfile an fill it up with the prepared
# filesystem structure from var/ram. The use of tar has the
# advantage that permissions will be set correctly. Then we run
# dhclient for all the other networking information and it runs
# dhclient-script and configures the variables. Then we mount
# /usr/, /opt, /tmp/users from our nfs server.
#
#
# the first part mounts a rw ram disk for devices etc.
#
###################################################################
#
. /etc/rc.config.default
. /etc/rc.status

echo "Running $0."
echo "Preparing Ramfilesystem"
# Decide wether to use the minix ramdisk or ramfs
mount ramfs || (mkfs.minix /dev/ram && mount /dev/ram) ||      \
rc_failed
rc_status -v1; rc_reset

# Show progress to users if device is present
test -w /proc/progress && echo 51 >/proc/progress

# Untarring filesystem into place
echo "Creating basic filesystem ..."
test -w /proc/progress && \
echo "52 Creating basic filesystem" >/proc/progress
mkdir /RAM/dev
mknod /RAM/dev/null c 1 3
tar  --exclude=*.tgz -xpzf /var/ram/ramdisk.tgz -C             \
/RAM >/dev/null 2>&1
rc_status -v1; rc_reset

# Mount local filesystems in '/etc/fstab'
echo "Mounting local file systems..."
test -w /proc/progress && \
echo "53 Mounting local file systems" >/proc/progress
mount -van >/dev/null 2>&1 || rc_failed
rc_status -v1; rc_reset


###################################################################
#
# start loopback networking
#
if test "$START_LOOPBACK" = yes; then
```

```
        echo "Setting up loopback device"
        ifconfig lo 127.0.0.1 netmask 255.0.0.0 up || rc_failed
        route add -net 127.0.0.0 netmask 255.0.0.0 dev lo ||   \
rc_failed
rc_status -v1; rc_reset
test -w /proc/progress && \
echo "55 Setting up loopback device"           \
 >/proc/progress
fi

##################################################################
#
# reading nfsserver from /proc/mounts and mounting /usr,
# /opt, /tmp/users
#
SERVER=‘cat /proc/mounts|grep " / " | \
sed -e 's!.*addr=\([^[:space:]]*\).*!\1!'‘
startproc /sbin/portmap || rc_failed
test -w /proc/progress && \
echo "56 Mounting Filesystems" >/proc/progress
mount -o rsize=8192,ro -t nfs $SERVER:/usr /usr ||     \
rc_failed
test -w /proc/progress && \
echo "57 ... usr ..." >/proc/progress
mount -o rsize=8192,ro -t nfs $SERVER:/opt /opt ||     \
rc_failed
test -w /proc/progress && \
echo "58 ... opt ..." >/proc/progress
mount -o rsize=8192,wsize=8192,rw -t nfs \
$SERVER:/TMPDSK/tmp/users /tmp/users    ||     \
rc_failed
test -w /proc/progress && \
echo "59 ... tmp ..." >/proc/progress

echo "Setting up NFS filesystem"
test -w /proc/progress && \
echo "60 Setting up NFS filesystem" >/proc/progress
rc_status -v1; rc_reset

##################################################################
#
# Set hostname, resolving, yp and other network parameters
# provided by (specified) dhcp server.
#
echo "Setting up network environment ..."
test -w /proc/progress && \
echo -n "62 Setting up network environment ..."        \
 >/proc/progress
#/sbin/dhclient -q -s $SERVER eth0 &>/dev/null || rc_failed
/sbin/dhclient -q eth0 &>/dev/null || rc_failed
rc_status -v1; rc_reset
```

```
if ! test -f /RAM/etc/rc.config ; then
cp /etc/rc.config.default /RAM/etc/rc.config
fi
. /etc/rc.config

##################################################################
#
# Start bootup client scripts.
#
if test -d /sbin/init.d/boot.d; then
        for i in /sbin/init.d/boot.d/S*; do
                test -f $i || continue
                echo running $i
                test -f "$i" && {
                    /bin/sh $i b || rc_failed
                }
                rc_status -v1; rc_reset
        done
fi
##################################################################
#
# start user defined bootup script.
#
if test -f /sbin/init.d/boot.local ; then
        ECHO_RETURN=$rc_done_up
echo "Running /sbin/init.d/boot.local"
        test -w /proc/progress &&                               \
echo "65 Running boot.local" >/proc/progress
        /bin/sh /sbin/init.d/boot.local || rc_failed
rc_status -v1; rc_reset
fi

##################################################################
#
# Let ld.so rebuild its cache.
#
test -x /sbin/ldconfig && {
echo "Setting up /etc/ld.so.cache"
test -w /proc/progress &&                                   \
echo -n "67 Setting up ld.so.cache"                    \
>/proc/progress
/sbin/ldconfig -X -C /RAM/etc/ld.so.cache                   \
>/dev/null || rc_failed
rc_status -v1; rc_reset
}

# Read all kernel messages generated until now and put them in
# one file.

test "$LOGGING" = no && exit 0
if test -x /usr/sbin/klogd ; then
        ECHO_RETURN=$rc_done_up
```

```
        echo "Creating /var/log/boot.msg"
test -w /proc/progress && echo                          \
"68 Creating file boot.msg" >/proc/progress
        rm -f /var/log/boot.msg
        /usr/sbin/klogd -f /var/log/boot.msg -o
        test -s /var/log/boot.msg || rc_failed_up
        rc_status -v1; rc_reset
fi

exit 0
```

D.2 Der "dhclient"-Prozess

D.2.1 Die Konfigurationsdatei *dhclient.conf*

Alle notwendigen Konfigurations- und Netzwerkparameter werden über das zentrale Instrument DHCP bereitgestellt. Die Anforderung notwendiger Variablen erfolgt mittels **dhclient**, welches seinerseits über die Datei *(/nfsroot/dxs)/etc/dhclient.conf* in seiner Funktion bestimmt wird. Die zusätzlich definierten benutzerspezifischen DHCP-Optionen werden analog zu den Einträgen in der Server-Datei */etc/dhcpd.conf* am Anfang der Datei angegeben.
Soll sich der Client mittels "vendor code identifier" beim DHCP-Server identifizieren, so läßt sich dieses durch das Senden des entsprechenden Strings[1] erreichen. Auf diese Weise kann beim Server entschieden werden, welche Konfiguration bei Bedarf zurückgeschickt wird. Das erlaubt gleichzeitig ausgewählte Variablen doppelt zu belegen, wie z.B. "filename" in Verwendung mit PXE und Etherboot im Ketten-Boot-Verfahren[2].
Das Timingverhalten beim Stellen der DHCP-Anfrage kann durch die Parameter "retry", "reboot", "select-time", "timeout" verändert werden, wobei jedoch die Default-Parameter sinnvolle Standardwerte einstellen, welche für die meisten Umgebungen gute Ergebnisse erzielen.

```
# dhclient configuration file
# see "man dhclient.conf" for further details
# file: /etc/dhclient.conf
#
# User options are defined like in dhcpd.conf. Set of options
# required for use with diskless X stations. Version 0.2.0
#
# modifications by Dirk von Suchodoletz <dirk@goe.net>, 10-06-2002
#
# -- user defined options --
```

[1] hier wurde als Beispiel "dxs" (62:79:73) verwendet
[2] siehe Abschnitt C.3.4

```
option o128 code 128              = string;
option o129 code 129              = string;
option menudflts code 160         = string;
option motdline1 code 184         = string;
option menuline1 code 192         = string;
option menuline2 code 193         = string;
option menuline3 code 194         = string;
option bootlocal-script code 221  = string;
option x-server-defs code 222     = string;
option start-x code 223           = string;
option start-snmp code 224        = string;
option start-sshd code 225        = string;
option start-xdmcp code 226       = string;
option start-cron code 227        = string;
option crontab-entries code 228   = string;
option start-rwhod code 229       = string;
option start-lpd code 230         = string;
option start-dnetc code 231       = string;
option tex-enable code 232        = string;
option netbios-workgroup code 233 = string;
option start-vmkernel code 234    = string;

# -- user defined defaults --
#
# defined within etc/dhcpd.conf of the server
# we need the defaults here for generating etc/rc.config
#
default start-x "indirect";
default start-snmp "yes";
default start-sshd "yes";
default start-dnetc "yes";
default start-xdmcp "yes";
default start-cron "yes";
default crontab-entries "";
default start-lpd "yes";

send dhcp-max-message-size 1024;
send dhcp-lease-time 72000;
#request subnet-mask, broadcast-address, time-offset,
# routers, domain-name, domain-name-servers,
# host-name, nis-servers,nis-domain, font-servers,
# ntp-servers, log-servers, x-display-manager,
# lpr-servers, start-sshd, start-dnetc, start-lpd,
# start-rwhod, crontab-entries, start-cron;
request;
#require start-xdmcp, domain-name-servers, x-server-defs, start-x;
require x-display-manager, nis-servers, nis-domain, font-servers,
ntp-servers, log-servers;
#initial-interval 1;
#timeout 1;
#retry 1;
#reboot 10;
```

```
script "/sbin/dhclient-script";
send vendor-class-identifier 64:79:73;
```

D.2.2 Das Konfigurationstool "dhclient-script"

An dieser Stelle wird nur das **dhclient-script** für DXS wiedergegeben, da dieses eine Obermenge der Konfigurationsmöglichkeiten darstellt. Für die Einstellungen von X-Terminals können einige Parameter entfallen. Dieses Skript verwendet die Unix-Shell **bash** und ist in seiner Funktion mit den Runlevel-Skripten zu vergleichen.
dhclient-script wird von **dhclient** über ein Temporärskript in */tmp* mit allen erhaltenen Variablen aufgerufen. Diese werden als Shell-Variablen in der aufrufenden Kommandozeile übergeben.

```
#!/bin/sh
#
# dhclient-script for Linux. Dan Halbert, March, 1997.
# Updated for Linux 2.[124] by Brian J. Murrell, January 1999.
# No guarantees about this. I'm a novice at the details of Linux
# networking.
#
# Modified for use with diskless X stations: Version 0.8.1b
# Dirk von Suchodoletz <dirk@goe.net>, 09-12-2001.

# Notes:

# 0. This script is based on the netbsd script supplied with
#  dhcp-970306.

# 1. ifconfig down apparently deletes all relevant routes and
# flushes the arp cache, so this doesn't need to be done
# explicitly.

# 2. The alias address handling here has not been tested AT ALL.
# I'm just going by the doc of modern Linux ip aliasing, which
# uses notations like eth0:0, eth0:1, for each alias.

# 3. I have to calculate the network address, and calculate the
# broadcast address if it is not supplied. This might be much
# more easily done by the dhclient C code, and passed on.

# 4. TIMEOUT not tested. ping has a flag I don't know, and I'm
# suspicious of the $1 in its args.

# 5. NIS, /etc/FONTSERVER, Xaccess, XF86Config, rc.config ...
# are setup by this script.

# 6. We start X with a trick via /etc/inittab while modifying
# the apropriate runlevel entry (telinit q).
```

```
# Must be used on exit. Invokes the local dhcp client exit
# hooks, if any.
function exit_with_hooks() {
  exit_status=$1
  if [ -x /etc/dhclient-exit-hooks ]; then
    . /etc/dhclient-exit-hooks
  fi
# probably should do something with exit status of the local
# script
  exit $exit_status
}

# Invoke the local dhcp client enter hooks, if they exist.
if [ -x /etc/dhclient-enter-hooks ]; then
  exit_status=0
  . /etc/dhclient-enter-hooks
  # allow the local script to abort processing of this state
  # local script must set exit_status variable to nonzero.
  if [ $exit_status -ne 0 ]; then
    exit $exit_status
  fi
fi

release=`uname -r`
release=`expr $release : '\(.*\)\..*'`
relmajor=`echo $release |sed -e 's/^\([^\.]*\)\..*$/\1/'`
relminor=`echo $release |sed -e 's/^.*\.\([^\.]*\)$/\1/'`

if [ x$new_broadcast_address != x ]; then
  new_broadcast_arg="broadcast $new_broadcast_address"
fi
if [ x$old_broadcast_address != x ]; then
  old_broadcast_arg="broadcast $old_broadcast_address"
fi
if [ x$new_subnet_mask != x ]; then
  new_subnet_arg="netmask $new_subnet_mask"
fi
if [ x$old_subnet_mask != x ]; then
  old_subnet_arg="netmask $old_subnet_mask"
fi
if [ x$alias_subnet_mask != x ]; then
  alias_subnet_arg="netmask $alias_subnet_mask"
fi

if [ x$reason = xMEDIUM ]; then
  # Linux doesn't do mediums (ok, ok, media).
  exit_with_hooks 0
fi

if [ x$reason = xPREINIT ]; then
  if [ x$alias_ip_address != x ]; then
    # Bring down alias interface. Its routes will disappear too.
```

```
    ifconfig $interface:0- inet 0
  fi
  if [ $relmajor -lt 2 ] || \
     ( [ $relmajor -eq 2 ] && [ $relminor -eq 0 ] ) ; then
    ifconfig $interface inet 0.0.0.0 netmask 0.0.0.0 \
broadcast 255.255.255.255 up
    # Add route to make broadcast work. Do not omit netmask.
    route add default dev $interface netmask 0.0.0.0
  else
    ifconfig $interface up
  fi

  # We need to give the kernel some time to get the interface up.
  sleep 1

  exit_with_hooks 0
fi

if [ x$reason = xARPCHECK ] || [ x$reason = xARPSEND ]; then
  exit_with_hooks 0
fi

if [ x$reason = xBOUND ] || [ x$reason = xRENEW ] || \
   [ x$reason = xREBIND ] || [ x$reason = xREBOOT ]; then
  if [ x$old_ip_address != x ] && [ x$alias_ip_address != x ] && \
[ x$alias_ip_address != x$old_ip_address ]; then
    # Possible new alias. Remove old alias.
    ifconfig $interface:0- inet 0
  fi
  if [ x$old_ip_address != x ] && \
     [ x$old_ip_address != x$new_ip_address ]; then
    # IP address changed. Bringing down the interface will delete
    # all routes, and clear the ARP cache.
    ifconfig $interface inet down

  fi
  if [ x$old_ip_address = x ] || \
     [ x$old_ip_address != x$new_ip_address ] || \
     [ x$reason = xBOUND ] || [ x$reason = xREBOOT ]; then

    ifconfig $interface inet $new_ip_address $new_subnet_arg    \
       $new_broadcast_arg
    # Add a network route to the computed network address.
    if [ $relmajor -lt 2 ] || \
       ( [ $relmajor -eq 2 ] && [ $relminor -eq 0 ] ); then
      route add -net $new_network_number $new_subnet_arg dev    \
       $interface
    fi
    for router in $new_routers; do
       route add default gw $router
    done
  fi
```

```
  if [ x$new_ip_address != x$alias_ip_address ] &&              \
     [ x$alias_ip_address != x ]; then
    ifconfig $interface:0- inet 0
    ifconfig $interface:0 inet $alias_ip_address                \
       $alias_subnet_arg
    route add -host $alias_ip_address $interface:0
  fi
  # set new hostname
  test -n "$new_host_name" && ( hostname $new_host_name;
echo >/etc/hosts
sed -e "s,#1.2.3.4*,$new_ip_address,g"                      \
    -e "s,default,$new_host_name,g" /etc/hosts.default      \
    >>/RAM/etc/hosts )
  # write font server ip to file
  test -n "$new_font_servers" && ( echo > /RAM/etc/FONTSERVER;  \
    for fontserver in $new_font_servers; do
        echo $new_font_servers >>/RAM/etc/FONTSERVER
    done )
  # configure logging
  test -n "$new_log_servers" && ( echo >/RAM/etc/syslog.conf;   \
    for logserver in $new_log_servers; do
        sed -e "s,#\*.\*;*\*.*,*.*;*kern.!*    @$logserver,g"   \
        /etc/syslog.conf.default >>/etc/syslog.conf
    done )
  # set up nis if defined
  test -n "$new_nis_domain" && domainname $new_nis_domain
  test -n "$new_nis_servers" && \
    echo ypserver $new_nis_servers >/etc/yp.conf
  # write available X display manager to var/lib/xdm/Xaccess
  test -n "$new_x_display_manager" && \
    echo -en "*\n%hostlist\tlocalhost " >/RAM/etc/Xaccess
    echo -e "$new_x_display_manager\n*\t\tCHOOSER %hostlist"    \
      >>/RAM/etc/Xaccess
# set up domainname and resolving
  test -n "$new_domain_name" && \
    echo search $new_domain_name >/etc/resolv.conf
  test -n "$new_domain_name_servers" && (
  # set up name resolving
  for nameserver in $new_domain_name_servers; do
    echo nameserver $nameserver >>/etc/resolv.conf
  done )
  # set up net time service
  test -n "$new_ntp_servers" && ( echo >/etc/ntp.conf;
    for ntpserver in $new_ntp_servers; do
        initialntp="$initialntp $ntpserver"
echo server $ntpserver >>/etc/ntp.conf
    done
    ntpdate -bs $initialntp &>/dev/null & )
  # set up printing environment
  test -n "$new_lpr_servers" && (
    echo -e "ServerName $new_lpr_servers">/etc/cups/client.conf
    lpstat -a -h $new_lpr_servers |                             \
```

```
    awk '{ print $1":"}' >/RAM/etc/printcap & )
  # set up X11 configuration
  XF86CFG=( $new_x_server_defs )
  test -n "$new_x_server_defs" && (
    for maxres in 640x480 800x600 1024x768 1280x1024 1600x1200
         do modes="\"$maxres\" \"lcd$maxres\" $modes"
            if [ $maxres == "${XF86CFG[5]}" ] ; then break ; fi
         done
    sed -e "s,LANG,${XF86CFG[0]},g" -e "s,MP,${XF86CFG[1]},g"  \
        -e "s,MD,${XF86CFG[2]},g"   -e "s,HS,${XF86CFG[3]},g"  \
        -e "s,VS,${XF86CFG[4]},g"   -e "s,MODES,$modes,g"      \
        -e "s,DRV,${XF86CFG[6]},g"  -e "s,CDP,${XF86CFG[7]},g" \
          /etc/X11/XF86Config.default >/RAM/etc/XF86Config
if [ $maxres == 1280x1024 ]     \
|| [ $maxres == 1600x1200 ] ; then sed -e                  \
"s,Modes[ ]*\"$maxres\" \"lcd$maxres\",Modes       ,g"     \
/RAM/etc/XF86Config >/RAM/etc/XF86Config.lowres ; fi
    ln -sf /dev/${XF86CFG[2]} /dev/mouse )
  # update inittab
  test -n "$new_host_name" && test -n "$new_start_x" &&        \
    ( echo >/RAM/etc/inittab
     if [ $new_start_x = "broadcast" ]; then
sed -e "s,#7:5.*,7:5:respawn:/usr/X11R6/bin/X vt1          \
 -$new_start_x >/dev/null 2>&1,g" /etc/inittab.default
 >>/RAM/etc/inittab
     else
sed -e "s,#7:5:.*,7:5:respawn:/usr/X11R6/bin/X vt1         \
 -$new_start_x $new_host_name \&>/dev/null,g"              \
 /etc/inittab.default >>/etc/inittab;# fi
     fi)

  # add commands to boot.local
  test -n "$new_bootlocal_script" && \
    echo $new_bootlocal_script >>/RAM/run/boot.local
  # set up /etc/rc.config (SuSE linux)
  # all parameters must be present, at least via defaults in
  # (nfsroot/dxs)/etc/dhclient.conf
  ( sed -e "s,KEYTABLE=.*,KEYTABLE=\"${XF86CFG[0]}\",i"        \
-e "s,START_X=.*,START_X=\"$new_start_x\",i"           \
        -e "s,START_SNMP.*,START_SNMPD=$new_start_snmp,i"      \
        -e "s,START_SSHD.*,START_SSHD=$new_start_sshd,i"       \
        -e "s,START_DNETC.*,START_DNETC=$new_start_dnetc,g"    \
        -e "s,DISPLAYM.*,DISPLAYMANAGER=$new_start_xdmcp,g"    \
        -e "s,CRON.*,CRON=$new_start_cron,g"                   \
        -e "s,START_RWHOD.*,START_RWHOD=$new_start_rwhod,g"    \
        -e "s,START_LPD.*,START_LPD=$new_start_lpd,g"          \
      /etc/rc.config.default | grep -v "#" >/RAM/etc/rc.config )
  # set up cron if started
  test "$new_start_cron" = yes && \
    ( cat /etc/crontab.default > /etc/crontab
    echo -e `echo -e "$new_crontab_entries"` >> /etc/crontab )
  exit_with_hooks 0
```

```
fi

if [ x$reason = xEXPIRE ] || [ x$reason = xFAIL ]; then
  if [ x$alias_ip_address != x ]; then
    # Turn off alias interface.
    ifconfig $interface:0- inet 0
  fi
  if [ x$old_ip_address != x ]; then
    # Shut down interface, which will delete routes and clear
    # arp cache.
    ifconfig $interface inet down
  fi
  if [ x$alias_ip_address != x ]; then
    ifconfig $interface:0 inet $alias_ip_address               \
      $alias_subnet_arg
    route add -host $alias_ip_address $interface:0
  fi
  exit_with_hooks 0
fi

if [ x$reason = xTIMEOUT ]; then
  if [ x$alias_ip_address != x ]; then
    ifconfig $interface:0- inet 0
  fi
  ifconfig $interface inet $new_ip_address $new_subnet_arg     \
           $new_broadcast_arg
  set $new_routers
  ############## what is -w in ping?
  if ping -q -c 1 $1; then
    if [ x$new_ip_address != x$alias_ip_address ] &&           \
[ x$alias_ip_address != x ]; then
      ifconfig $interface:0 inet $alias_ip_address             \
               $alias_subnet_arg
      route add -host $alias_ip_address dev $interface:0
    fi
    if [ $relmajor -lt 2 ] || \
( [ $relmajor -eq 2 ] && [ $relminor -eq 0 ] );
 then route add -net $new_network_number
    fi
    for router in $new_routers; do
      route add default gw $router
    done
    exit_with_hooks 0
  fi
  ifconfig $interface inet down
  exit_with_hooks 1
fi

exit_with_hooks 0
```

D.2.3 Die XFree86 Konfigurationsdatei

Die Datei *(/nfsroot/dxs)/etc/X11/XF86Config.default* wird von **dhclient-script** interpretiert und mittels **sed** die Konfigurationsdatei in */RAM/etc/X11/XF86Config* angelegt. An dieser Stelle wird nur die Variante für XFree 4.X.Y vorgestellt, da sich die Version für XFree 3.3.6 leicht ableiten läßt. Für das "alte" XFree ist es noch notwendig den entsprechenden Link auf den Grafikserver durch **dhclient-script** anlegen zu lassen. Diese Datei soll als Beispiel und Vorlage für ähnliche Anwendungen dienen. Mit der *rc.config.default* wird analog verfahren.

```
Section "Module"
    Load        "dbe"
    SubSection  "extmod"
      Option    "omit xfree86-dga"
    EndSubSection
    Load        "type1"
    Load        "speedo"
    Load        "freetype"
EndSection
Section "Files"
    RgbPath    "/usr/X11R6/lib/X11/rgb"
    FontPath   "/usr/X11R6/lib/X11/fonts/misc/:unscaled"
    FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
    FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
    FontPath   "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath   "/usr/X11R6/lib/X11/fonts/URW/"
    FontPath   "/usr/X11R6/lib/X11/fonts/Speedo/"
    FontPath   "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath   "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath   "/usr/X11R6/lib/X11/fonts/100dpi/"
    FontPath   "/usr/X11R6/lib/X11/fonts/PEX/"
    FontPath   "/usr/X11R6/lib/X11/fonts/CID/"
    ModulePath "/usr/X11R6/lib/modules"
EndSection
Section "InputDevice"
    Identifier "Keyboard1"
    Driver "keyboard"
    Option  "XkbRules" "xfree86"
    Option  "XkbLayout" "LANG"
    Option  "XkbModel" "pc102"
    Option  "XkbVariant" "nodeadkeys"
EndSection
Section "InputDevice"
    Identifier "Mouse1"
    Driver "mouse"
    Option "Protocol" "MP"
    Option "Device" "/dev/MD"
    Option "Emulate3Buttons"
    Option "ZAxisMapping"  "4 5"
    Option "Buttons"       "3"
```

```
EndSection
Section "Monitor"
    Identifier "Default"
    HorizSync   31.5 - HS
    VertRefresh 50 - VS
    Option     "DPMS"
Modeline "640x480"    31.50  640  680  720  864  480  488  491  521
Modeline "640x480"    45.80  640  672  768  864  480  488  494  530
Modeline "800x600"    50.00  800  856  976 1040  600  637  643  666
Modeline "800x600"    69.65  800  864  928 1088  600  604  610  640
Modeline "1024x768"   80.00 1024 1052 1164 1360  768  784  787  823
Modeline "1024x768"   86.00 1024 1040 1152 1360  768  784  787  823
Modeline "1024x768"   98.90 1024 1056 1216 1408  768  782  788  822
Modeline "1024x768"  115.50 1024 1056 1248 1440  768  771  781  802
Modeline "1152x864"   92.00 1152 1208 1368 1474  864  865  875  895
Modeline "1152x864"  110.00 1152 1240 1324 1552  864  864  876  908
Modeline "1280x1024" 145.00 1280 1312 1456 1712 1024 1027 1030 1064
Modeline "1280x1024" 157.50 1280 1344 1504 1728 1024 1025 1028 1072
Modeline "1600x1200" 202.50 1600 1664 1856 2160 1200 1201 1204 1250
EndSection
Section "Device"
    Identifier  "StdGraphics"
    Option      "power_saver"
    Driver      "DRV"
EndSection
Section "Screen"
    Identifier  "Screen 1"
    Device      "StdGraphics"
    Monitor     "Default"
    DefaultColorDepth CDP
    Subsection "Display"
        Depth       8
        Modes       MODES
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       16
        Modes       MODES
        ViewPort    0 0
    EndSubsection
    Subsection "Display"
        Depth       24
        Modes       MODES
        ViewPort    0 0
    EndSubsection
EndSection
Section "ServerLayout"
    Identifier  "Simple Layout"
    Screen  "Screen 1"
    InputDevice "Mouse1" "CorePointer"
    InputDevice "Keyboard1" "CoreKeyboard"
EndSection
```

D.3 Skript zur Erzeugung der */etc/dhcpd.conf*

Das nachfolgend angegebenes Skript (**dhcp-generate**) erstellt die *(/var/lib/dhcp)/etc/dhcpd.conf* mittels entsprechender Datenbankabfrage. Die Datenbankzugriffsparameter sind zu Beginn des Skriptes definiert. Es wäre jedoch auch vorstellbar diese über Kommandozeilenoptionen zu überge-ben, wenn das Tool generischer eingesetzt werden bzw. die Sicherheit der Daten erhöht werden soll. Einige Standardeinstellungen sind an die hier vorgestellte Beispielumgebung angepaßt und würden in anderen Umfeldern sicherlich anders gesetzt werden müssen. Sie erleichtern jedoch den Umgang mit fehlenden oder unvollständigen Datenbankeinträgen. Möchte man hier näher informiert werden, sei auf das Report-Tool in Abschnitt E.1 verwiesen.

```
#!/usr/bin/perl
#
# Skript zur Generierung der /etc/dhcpd.conf, Version 0.9.2e
#
# Dirk von Suchodoletz <dirk@goe.net>, 25-10-2001
#
##################################################################
# Subroutinen zur Typumwandlung
sub btodq {
        my $dq = join ".",unpack("CCCC",$_[0]);
        return $dq;
}
sub dqtob {
        my @dq;
        my $q;
        my $i;
        my $bin;

        foreach $q (split /\./,$_[0]) {
                push @dq,$q;
        }
        for ($i = 0; $i < 4 ; $i++) {
                if (! defined $dq[$i]) {
                        push @dq,0;
                }
        }
        $bin    = pack("CCCC",@dq);      # 4 unsigned chars
        return $bin;
}

# Auswerten der Kommandozeilenparameter
$num=0;
foreach ( @ARGV ) {
# Interfacedeklaration
    if (/^-i/) { }
elsif (/[^-]/ && /eth/) {
$if[$num]=$_;
```

```
$num++; }
}
if (!$if[0]) {$if[0]="eth0";}

# Verbindung zur Datenbank
use Mysql;
$DB = Mysql->connect(s15, Hardwareverwaltung, hwvdb);

# Zieldatei oeffnen
open (OUT, ">/etc/dhcpd.conf.new");
open (ERR, ">/var/log/dhcpdconf.log");

# Statischer Header der /etc/dhcpd.conf
print OUT "
# /etc/dhcpd.conf
#
# Configuration file for ISC dhcpd
#
# Automatically generated by dhcp-generate.pl\n#\n#";
($sec,$min,$hour,$mday,$mon,$year,
  $wday,$yday,$isday) = localtime (time);
print OUT " --> ",$year+1900,"/",$mon+1,"/",$mday+1;
print OUT " $hour:$min";
print OUT "
#
# (c) Dirk von Suchodoletz <dirk\@goe.net>, 2001
#
# -- user defined vendor options --

option o128 code 128             = string;
option o129 code 129             = string;
option menudflts code 160        = string;
option motdline1 code 184        = string;
option menuline1 code 192        = string;
option menuline2 code 193        = string;
option menuline3 code 194        = string;
option bootlocal-script code 221 = string;
option x-server-defs code 222    = string;
option start-x code 223          = string;
option start-snmp code 224       = string;
option start-sshd code 225       = string;
option start-xdmcp code 226      = string;
option start-cron code 227       = string;
option crontab-entries code 228  = string;
option start-rwhod code 229      = string;
option start-lpd code 230        = string;
option start-dnetc code 231      = string;

# -- global options --

option o128                        E4:45:74:68:00:00;
deny                               unknown-clients;
```

```
default-lease-time                160000;
max-lease-time                    200000;
use-host-decl-names               on;
option dhcp-max-message-size      1024;
ddns-update-style                 none;

# description for several options:
# A. x-server-defs \"de PS/2 psaux 96 100 1280x1024 nv 16\"
#  - Sets several values for XFree86 V 4.0.X
#
#  1. keyboard  de/us
#  2. mouse typ PS/2/MouseMan/MicroSoft/...
#  3. mouse device psaux/ttyS0X
#  4. horiz. freq. monitor 40 - 96 (depending on monitor)
#  5. vert. freq. monitor 56 -120  (depending on monitor)
#  6. max. resolution
#  6. graphic server module  mga/i810/nv/ati/...
#        (depends on hardware)
#  7. display color depth 8/16/24

# -- vendor identifier dependend settings --
class \"Etherboot\" {
    match if substring (option vendor-class-identifier, 0, 9) =
\"Etherboot\";
        option motdline1  = \"Welcome to Guru Labs classroom\";
        option vendor-encapsulated-options
3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
class \"EB-3c905c\" {
    match if substring (option vendor-class-identifier, 0, 9) =
\"EB-3c905c\";
#       option menudflts  = \"timeout=30:default=193\";
        option motdline1  = \"Welcome to Juridicum CIP-Pool\";
        option menuline1  = \"Anmelden an CIP (CIP-Pool der Jur.
Fakultaet):::/dev/hda:::\";
        option menuline2  = \"Anmelden an STUD (Internet Hotline)
:::/nfsroot/dxs/boot/wsbootimg:::\";
#       option menuline3  = \"Boot from Floppy:::/dev/fd0:::\";
        option vendor-encapsulated-options
3c:09:53:74:75:64:69:6e:65:74:7a:ff;
}
class \"SUB-XOpac\" {
    match if substring (option vendor-class-identifier, 0, 9) =
\"SUB-XOpac\";
option motdline1  = \"Welcome to SUB-OPAC\";
        option vendor-encapsulated-options 3c:09:53:55:42:ff;
}
class \"PXEClient:\" {
    match if substring (option vendor-class-identifier, 0, 10) =
\"PXEClient:\";
        filename \"/nfsroot/dxs/boot/3c905c-tpo.pxe\";
}
```

```
# -- client specific --\n\n";

################################################################
# Bestimmen der IP des Bootservers  auf den einzelnen Interfaces
# und setzen der entsprechenden Netzwerkdaten

$num=0;
while ($if[$num]) {
$iface=$if[$num];
$ipcfg = `/sbin/ifconfig $iface` ;
@ipcfg = split  /^/, $ipcfg ;

foreach $i (@ipcfg) {
$_ = $i ;
if ( /Bcast/ ) {
  @i = split ;
  @j = split ( /\:/, $i[1] ) ;
   $ip = $j[1] ;
  @j = split ( /\:/, $i[2] ) ;
   $broadc = $j[1] ;
  @j = split ( /\:/, $i[3] ) ;
   $netm = $j[1] ; }
}

$net = btodq ( dqtob( $ip ) & dqtob( $netm ) );

# Schreibe Subnetdeklaration
print OUT "subnet $net netmask $netm {\n    server-identifier";
print OUT " $ip;\n}\n";

# Gruppiere Rechner und lese Basisdaten
$sth01=$DB->query("select Rechner.GroupingID,
  HostName, IPAddress, RechnerID from Rechner,
  Grouping_Properties
  where Rechner.GroupingID=Grouping_Properties.GroupingID
  AND PropertyNameID=34 AND Value='$ip' order by
  Grouping_Properties.GroupingID, HostName;");
$old_grid=$new_grid=0;
while ( @data = $sth01->fetchrow )
{
   $new_grid=$data[0];
   if ( $old_grid != $new_grid ) {
  if ( $old_grid != 0 ) { print OUT "}\n"; }
# Auslesen der Daten der Gruppe des Rechners
  $sth02=$DB->query("SELECT FirstDNS, SecondDNS, DomainName,
  Broadcast, GateWay, NetMask, Name, RechnerArtID FROM
  Grouping WHERE GroupingID=$new_grid");
  @data02 = $sth02->fetchrow;
  if ($data02[0] ne '') { $domain_name_servers=$data02[0]; }
  if ($data02[1] ne '')
   { $domain_name_servers.=', '.$data02[1]; }
  if ($data02[2] ne '') { $domain_name=$data02[2]; }
```

```
  if ($data02[3] ne '') { $broadcast=$data02[3]; }
  if ($data02[4] ne '') { $router=$data02[4]; }
  if ($data02[7] eq '1')
   { $filename="/nfsroot/dxs/boot/wsbootimg";
     $rootpath="/nfsroot/dxs"; }
  else { $filename="/nfsroot/dxt/boot/bootimg";
        $rootpath="/nfsroot/dxt"; }


# Ermitteln der weiteren Gruppendaten
  $sth03=$DB->query("SELECT Grouping_Properties.Value,
  PropertyNamen.Name FROM Grouping_Properties, PropertyNamen
  WHERE Grouping_Properties.GroupingID='$new_grid' AND
  Grouping_Properties.PropertyNameID=
   PropertyNamen.PropertyNameID");

  while ( @add_data = $sth03->fetchrow )
  {
SWITCH: {
($add_data[1] eq 'XDM') && do
{ $start_xdmcp=$add_data[0]; last SWITCH; };
($add_data[1] eq 'RWHO') && do
{ $start_rwho=$add_data[0]; last SWITCH; };
($add_data[1] eq 'CRON') && do
{ $start_cron=$add_data[0]; last SWITCH; };
($add_data[1] eq 'X11') && do
{ $start_x11=$add_data[0]; last SWITCH; };
($add_data[1] eq 'SNMP') && do
{ $start_snmp=$add_data[0]; last SWITCH; };
($add_data[1] eq 'DNETC') && do
{ $start_dnetc=$add_data[0]; last SWITCH; };
($add_data[1] eq 'NIS-Domain') && do
{ $nis_domain=$add_data[0]; last SWITCH; };
($add_data[1] eq 'X-Login-Server' ) && do {
if ( $x_display_manager eq '' )
{ $x_display_manager=$add_data[0]; }
else
{ $x_display_manager.=', '.$add_data[0]; }
last SWITCH; };
( $add_data[1] eq 'NTP-Server' ) && do {
if ($ntp_servers eq '')
{ $ntp_servers=$add_data[0]; }
else
{ $ntp_servers.=', '.$add_data[0]; }
last SWITCH; };
($add_data[1] eq 'Log-Server') && do {
if ($log_servers eq '' )
{ $log_servers=$add_data[0]; }
else
{ $log_servers.=', '.$add_data[0]; }
last SWITCH; };
($add_data[1] eq 'LPR-Server' ) && do {
```

```
if ($lpr_servers eq '')
{ $lpr_servers=$add_data[0]; }
else
{ $lpr_servers.=', '.$add_data[0]; }
last SWITCH; };
($add_data[1] eq 'Font-Server') && do {
if ($font_servers eq '')
{ $font_servers=$add_data[0]; }
else
{ $font_servers.=', '.$add_data[0]; }
last SWITCH; };
($add_data[1] eq 'DomainSearch') && do {
if ( $domain_name eq '' )
{ $domain_name=$add_data[0]; }
else
{ $domain_name.=' '.$add_data[0]; }
last SWITCH; };
($add_data[1] eq 'NIS-Server') && do {
if ($nis_servers eq '')
{ $nis_servers=$add_data[0]; }
else
{ $nis_servers.=', '.$add_data[0]; }
last SWITCH; };
}
}
print OUT "#\n# Rechner -> $data02[6]\n#\n";
print OUT "group {\n";
    print OUT "    filename \"$filename\";\n";
print OUT "    option root-path \"$rootpath\";\n";
if ( $broadcast ne '' ) {
print OUT "    option broadcast-address $broadcast;\n"; }
else {
   print ERR "Warn: Gruppe $data02[6] kennt keine";
   print ERR "Broadcast-Adresse\n"; }
if ( $router ne '' ) {
print OUT "    option routers $router;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine";
print ERR "Gateway-Adresse\n"; }
if ( $domain_name_servers ne '' ) {
print OUT "    option domain-name-servers"
print OUT "$domain_name_servers;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine";
print ERR "DNS-Adresse(n)\n"; }
if ( $domain_name ne '' ) {
print OUT "    option domain-name \"$domain_name\";\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt keine";
print ERR "Domain(s)\n"; }
if ( $nis_domain ne '' ) {
print OUT "    option nis-domain \"$nis_domain\";\n"; }
```

```
else {
    print ERR "Info: Gruppe $data02[6] kennt keine";
print ERR "NIS-Domain\n"; }
if ( $nis_servers ne '' ) {
print OUT "    option nis-servers $nis_servers;\n"; }
else {
   if ( $nis_domain ne '' ) {
  print ERR "Crit: Gruppe $data02[6] kennt keine";
  print ERR "NIS-Server  hat aber die \n";
      print ERR "NIS-Domain $nis_domain definiert\n"; }
   else {
  print ERR "Info: Gruppe $data02[6] kennt keine";
  print ERR "NIS-Server\n"; } }
if ( $log_servers ne '' ) {
print OUT "    option log-servers $log_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine";
print ERR "Log-Server\n"; }
if ( $ntp_servers ne '' ) {
print OUT "    option ntp-servers $ntp_servers;\n"; }
        else {
            print ERR "Info: Gruppe $data02[6] kennt keine";
print ERR "NTP-Server\n"; }
if ( $lpr_servers ne '' ) {
print OUT "    option lpr-servers $lpr_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine";
print ERR "LPR-Server (Printserver)\n"; }
if ( $font_servers ne '' ) {
print OUT "    option font-servers $font_servers;\n"; }
else {
    print ERR "Info: Gruppe $data02[6] kennt keine";
print ERR "Font-Server\n"; }
if ( $x_display_manager ne '' ) {
print OUT "    option x-display-manager";
print OUT "$x_display_manager;\n"; }
else {
    print ERR "Warn: Gruppe $data02[6] kennt XDMCP-Server\n"; }
if ( $start_x11 ne '' ) {
print OUT "    option start-x \"$start_x11\";\n"; }
else {
    print ERR "Warn: In Gruppe $data02[6] fehlt der";
print ERR "X-Zugriffsmodus\n"; }
if ( $start_xdmcp ne '' ) {
print OUT "    option start-xdmcp \"$start_xdmcp\";\n"; }
else {
    print ERR "Info: In Gruppe $data02[6] wird auf xdmcp"
print ERR "verzichtet\n"; }
print OUT "    option start-rwhod \"$start_rwho\";\n";
print OUT "    option start-cron \"$start_cron\";\n";
print OUT "    option start-snmp \"$start_snmp\";\n";
print OUT "    option start-dnetc \"$start_dnetc\";\n    #\n";
```

```
# Zuruecksetzen der Variablen
$nis_domain=$nis_servers=$font_servers=$log_servers='';
$x_display_manager=$boot_local=$mac=$ntp_servers='';
$start_dnetc=$start_snmp=$start_cron=$start_rwho='';
$start_xdmcp='';
  }
################################################################
# Ermitteln der Tastatursprache
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3]
  AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID
  AND HardWareNamen.HardWareArtID=3 AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=53");
if ( @add_data = $sth2->fetchrow ) {
   $key_lang = $add_data[0]; }
else {
    print ERR "Warn: Setze fuer $data[1] Tastatur-";
print ERR "Default-Sprache auf Deutsch\n";
$key_lang = "de"; }

# Ermitteln des notwendigen Mausprotokolls
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3]
  AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID
  AND HardWareNamen.HardWareArtID=2 AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=56");
if ( @add_data = $sth2->fetchrow ) {
   $mp = $add_data[0]; }
else {
    print ERR "Crit: Setze fuer $data[1] Maus-Protokoll";
print ERR "auf PS/2 da nicht in Datenbank vermerkt\n";
$mp = "PS/2"; }

# Ermitteln des notwendigen Mausanschlusses
$sth2=$DB->query("SELECT PropertyNamen.Name FROM
  HardWareNamen_Properties, HardWareNamen, HardWare,
  PropertyNamen WHERE PropertyNamen.PropertyArtID=3 AND
  PropertyNamen.PropertyNameID=
  HardWareNamen_Properties.HardWareNamen_PropertyNameID AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen.HardWareArtID=2 and HardWare.RechnerID=$data[3]
  AND HardWare.HardWareNameID=HardWareNamen.HardWareNameID");
if ( @add_data = $sth2->fetchrow ) {
  $_ = $add_data[0];
  if (/Serial/) { $md = "ttyS0"; }
  else { $md = "psaux"; } }
else {
```

```
    print ERR "Crit: Setze fuer $data[1] Maus-Anschluss";
print ERR "auf psaux da nicht in Datenbank vermerkt\n";
    $md = "psaux"; }

# Ermitteln der Monitordaten
# 1) max. Aufloesung
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
  HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
      HardWareNamen.HardWareArtID=1 AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=2");
if ( @add_data = $sth2->fetchrow ) {
   $max_res = $add_data[0]; }
else {
    print ERR "Warn: Setze fuer $data[1] Default-Monitor-";
print ERR "Aufloesung auf 1024x768 da nicht in Datenbank\n";
$max_res = "1024x768"; }

# 2) max. Vertikalfreq.
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
  HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
      HardWareNamen.HardWareArtID=1 AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=13");
if ( @add_data = $sth2->fetchrow ) {
   $vfreq = $add_data[0]; }
else {
print ERR "Crit: Nehme fuer $data[1] Monitor Vertikalfreq.";
print ERR "von 65kHz da nicht in Datenbank\n";
    $vfreq = "65"; }

# 3) max. Vertikalfreq.
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3]
  AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
      HardWareNamen.HardWareArtID=1 AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=14");
if ( @add_data = $sth2->fetchrow ) {
   $hfreq = $add_data[0]; }
else {
    print ERR "Crit: Nehme fuer $data[1] Monitor Horizontal";
print ERR "freq. von 90Hz, da nicht in Datenbank\n";
$hfreq = "90"; }

# 4) Farbtiefe (ergibt sich aus der Grösse des Grafikspeichers)
#    (Hier sollte eine Formel stehen, inzwischen sind 16bit aber
```

```
#     ohne Probleme zu erreichen)
$color_d="16";

# 5) XFree-Treiber (Version 4.0.X)
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3] AND
  HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
  (HardWareNamen.HardWareArtID=5 OR
  HardWareNamen.HardWareArtID= 23) AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=10");
if ( @add_data = $sth2->fetchrow ) {
       $driver = $add_data[0]; }
else {
$sth2=$DB->query("SELECT Value FROM HardWare, HardWareNamen,
  HardWareNamen_Properties WHERE HardWare.RechnerID=$data[3]
  AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
  (HardWareNamen.HardWareArtID=5 OR
  HardWareNamen.HardWareArtID= 23) AND
  HardWareNamen_Properties.HardWareNameID=
   HardWareNamen.HardWareNameID AND
  HardWareNamen_Properties.HardWareNamen_PropertyNameID=11");
if ( @add_data = $sth2->fetchrow ) {
       $driver = $add_data[0]; }
else {
print ERR "Warn: Setze fuer $data[1] XFree86-Modul auf";
print ERR "'vesa' (generic)\n";
    $driver = 'vesa'; }
}
# Ermitteln der MAC-Adresse
$sth2= $DB->query("SELECT Value FROM HardWare,
  HardWare_Properties, HardWareNamen WHERE
  HardWare.RechnerID=$data[3] AND
  HardWare.HardWareID=HardWare_Properties.HardWareID AND
  HardWare.HardWareNameID=HardWareNamen.HardWareNameID AND
  HardWare_Properties.PropertyNameID=44");
if ( @add_data = $sth2->fetchrow ) {
$mac = $add_data[0]; }
else {
    print ERR "Crit: Keine Mac-Adresse fuer Netzwerkkarte in";
print ERR "$data[1] ermittelbar\n";
    $mac = "00:00:00:00:00:00"; }

# Ermitteln der notwendigen Kernelmodule
$sth2=$DB->query("SELECT Value FROM HardWare,
  HardWareNamen_Properties WHERE HardWare.RechnerID='$data[3]' AND
  HardWareNamen_Properties.HardWareNameID=HardWare.HardWareNameID
  AND HardWareNamen_Properties.HardWareNamen_PropertyNameID='15'");

while ( @add_data = $sth2->fetchrow )
{
```

```
if ( $boot_local eq '' ) {
    $boot_local='modprobe -qa '.$add_data[0]; }
else {
    $boot_local.=' '.$add_data[0]; }
}

# Füllen des Hostteils
print OUT "    host $data[1] {\n";
print OUT "\thardware ethernet $mac;\n";
print OUT "\toption x-server-defs \"$key_lang $mp $md $vfreq ";
print OUT "$hfreq $max_res $driver $color_d\";\n";
print OUT "\toption bootlocal-script \"$boot_local\";\n";
# Framebuffer & LPP bei bestimmten Grafikkartentypen einschalten
if ($driver eq 'ati' || $driver eq 'nv' || $driver eq 'XF86_S3')
    { print OUT "\toption o129 \"vga=0x0301\";\n"; }
print OUT "\tfixed-address $data[2];\n    }\n";

# Zuruecksetzen der Variablen
$key_lang=$mac=$mp=$md=$vfreq=$hfreq=$max_res=$driver="";
$color_d=$boot_local="";

#################################################################
# Gruppenzugehoerigkeit
   $old_grid=$new_grid;
}
# Schliessende Klammer der letzten Gruppe
if ($old_grid) {print OUT "}\n";}
# Schliessende Klammer der Interface-Schleife
$num++;
}

# Schliesse Dateien
close (OUT);
close (ERR);
```

D.4 Steuerung des "dhcp-generate"

Mittels des Skripts **dhcp-cron** wird dafür gesorgt, dass mittels **dhcp-generate** eine neue Konfigurationsdatei für den **dhcpd** geschrieben wird *(/var/lib/dhcp)/etc/dhcpd.conf.new*. Dieses Skript startet man am besten zeitgesteuert mittels des Cron-Daemon. Dazu fügt man einen Eintrag in der */etc/crontab* hinzu oder legt ein entsprechendes Startskript unter */etc/cron.d* ab. Die neu generierte Konfigurationsdatei wird zuerst durch den testweisen Aufruf des DHCP-Servers auf einem anderen Port als dem BOOTPS (hier Port 66) auf syntaktische Richtigkeit überprüft. Konnte die Korrektheit verifiziert werden, wird die Datei in die Zieldatei umkopiert. Anschließend wird sichergestellt, dass die Dienste NFS (damit das Rootfilesystem zur Verfügung steht) und TFTPD (zur Übertragung des Kernels bzw. des

PXE-Images) korrekt funktionieren. Nun wird der eigentliche DHCP-Server mit der neuen Konfigurationsdatei erneut gestartet.

```
#! /bin/sh
#
# Script for Reconfiguration of dhcpd.conf and checking for
# sane working environment, Version 0.3.0a
#
# Copyright (c) 2001
#
# Author: Dirk von Suchodoletz <dirk@goe.net>, 2001
#
################################################################
. /etc/rc.config
/usr/local/bin/dhcp-generate -i $DHCPD_INTERFACE || exit 1
dhcpd -p 66 -cf /etc/dhcpd.conf.new || exit 1
mv /var/lib/dhcp/etc/dhcpd.conf \
/var/lib/dhcp/etc/dhcpd.conf.bak || exit 1
mv /etc/dhcpd.conf.new /var/lib/dhcp/etc/dhcpd.conf || exit 1
rcdhcpd stop &>/dev/null
mount -t nfs $HOSTNAME.goe.net:/nfsroot/dxs /var/adm/mount && \
umount /var/adm/mount || exit 1
checkproc tftpd || ( /etc/init.d/tftpd start && \
checkproc tftpd || exit 1 )
rcdhcpd start &>/dev/null
```

D.5 Skript zur Ersteinrichtung der Datenbank

In einigen Fällen hat man bereits eine Terminalkonfiguration auf Basis der */etc/dhcpd.conf* wie in Teil II. besprochen erstellt und möchte nun möglichst einfach migrieren. Nun liegt es nahe, nicht alle Daten komplett neu in die Datenbank einzupflegen, sondern einige bereits bekannte Eigenschaften der Thin-Clients aus der vorliegenden */etc/dhcpd.conf* abzuleiten. Folgendes Skript kann dabei vielleicht als Anhaltspunkt dienen. Einige Daten lassen sich dabei aus den Konfigurationsfeldern ableiten, bestimmte Eigenschaften, wie die MAC-Adresse[3] liefern darüberhinaus recht genaue Hinweise zur Netzwerkkarte. Die gezeigten Definitionen stammen aus der hier beschriebenen Problemstellung.

```
#!/usr/bin/perl
#
# Skript zum Generieren von Datenbankeinträgen ausgehend von der
# /etc/dhcpd.conf (Erstfuellung einer DB zur Migration der Daten)
#
# Dirk von Suchodoletz <dirk@goe.net>, 06-01-2001
#
#####################################################################
```

[3]Man kann sich auf den Seiten der FCC darüber informieren, welchen Herstellern, welche Adressbereiche des MAC-Adressierungsschemas zugeordnet sind. Dieses bilden bereits einige Netzwerkdiagnosetools in ihren Ausgaben ab.

```
use Mysql;

$DB = Mysql->connect(s8, Hardwareverwaltung, hwvadm, hwvadm1);

open (IN, "dhcpd.conf");


foreach (<IN>) {

# Erfassen der wesentlichen Eintraege (Hostname, MAC,
# IP-Nummer)

if (/host\s/) {
@a=/host\s([a-z0-9\-]{4,})\s/;
$hostname=$a[0];}
if (/ethernet/) {
@a=/\s([0-9A-F:]{17})/;
$MAC=$a[0];}
if (/ed-address/) {
@a=/\s([0-9\.]{9,})/;
$IP=$a[0];
    print "HOST $hostname, MAC $MAC, IP $IP\n";

# Defaultwerte fuer Beschaffung, Garantie und Produktnamen

$Beschafft="1998-10-15";
$Garantiebis="1999-10-14";
$HWNameID=18;
$Liefer=1;

$_=$MAC;

# Identifikation einzelner Systeme an ihren MAC-Adressen

if (/00:E0:7D:8/) {
    $Beschafft="2000-09-20";
$Garantiebis="2001-09-19";
$HWNameID=16;
}
if (/00:00:1C:DC/) {
$Beschafft="2000-07-20";
$Garantiebis="2001-07-19";
$HWNameID=17;
$Liefer=6;
}
if (/00:E0:4C/) {
    $Beschafft="2000-02-14";
$Garantiebis="2001-02-13";
$HWNameID=17;
}
if (/00:E0:7D:7/ || /00:E0:7D:0/) {
    $Beschafft="1999-09-20";
```

```
$Garantiebis="2000-09-19";
$HWNameID=18;
}
if (/00:E0:7D:7B/) {
    $Beschafft="2000-03-20";
$Garantiebis="2001-03-19";
$HWNameID=17;
}
if (/00:00:B4:/ || /00:4F:/) {
    $Beschafft="1997-01-01";
$Garantiebis="1998-01-01";
$HWNameID=81;
$Liefer=3;
}
if (/00:20:/) {
    $Beschafft="1996-01-01";
$Garantiebis="1996-01-01";
$Liefer=2;
$HWNameID=82;
}
if (/00:00:C0:/) {
    $Beschafft="1997-01-01";
$Garantiebis="1998-01-01";
$Liefer=7;
$HWNameID=101;
}

if (/00:60:/ || /00:A0:/) {
    $Beschafft="1998-10-10";
$Garantiebis="1999-10-09";
$Liefer=2;
$HWNameID=83;
}

$sth=$DB->query("SELECT RechnerID FROM Rechner
  WHERE HostName='$hostname'");
@data = $sth->fetchrow;

if ($#data==-1 ) {
$sth=$DB->query("insert into Rechner VALUES
  ('','$IP','$hostname',1)");
$sth=$DB->query("SELECT RechnerID FROM Rechner
  WHERE HostName='$hostname'");
@data = $sth->fetchrow;
        }
$RechnerID=$data[0];

print "RechID: $RechnerID\n";

$sth=$DB->query("SELECT RechnerPropertyID FROM
  Rechner_Properties WHERE RechnerID='$RechnerID'");
@data = $sth->fetchrow;
```

```
# if ($#data==-1 ) {
#     $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','46','stud.local')");
#     $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','46','gwdg.de')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','46','goe.net')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','34','134.76.60.29')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','34','134.76.60.25')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','34','134.76.60.26')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','35','134.76.60.67')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','47','134.76.60.67')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','48','134.76.60.23')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','49','ntps1.gwdg.de')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','49','ntps2.gwdg.de')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','49','ntps3.gwdg.de')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s4')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s5')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s6')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s9')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s10')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s11')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s12')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','s13')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','10.8.100.1')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','50','his.stud.local')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','52','134.76.60.28')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','51','chnsmi20')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','29','yes')");
# $sth=$DB->query("insert into Rechner_Properties \
```

```
#            VALUES ('','$RechnerID','30','yes')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','31','yes')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','32','indirect')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','24','yes')");
# $sth=$DB->query("insert into Rechner_Properties \
#            VALUES ('','$RechnerID','26','yes')");
#
# }
$sth=$DB->query("SELECT HardWareID FROM
  HardWare_Properties WHERE Value='$MAC'");
@data = $sth->fetchrow;



if ($#data==-1 ) {
$sth=$DB->query("SELECT * FROM HardWare WHERE \
                        RechnerID='$RechnerID'");
@data = $sth->fetchrow;
if ($#data==-1 ) {
    $sth=$DB->query("insert into HardWare VALUES \
                ('','$HWNameID','$RechnerID','$Liefer',\
                 '$Garantiebis','050005','00.00','$Beschafft',\
                 'Internet-Hotline')");
$sth=$DB->query("SELECT HardwareID FROM HardWare \
                 WHERE RechnerID='$RechnerID'");
@data = $sth->fetchrow;
         }
$HardwareID=$data[0];


$sth=$DB->query("insert into HardWare_Properties VALUES \
                ('','$HardwareID','44','$MAC')");
}
}
}
```

D.6 Skript zur Generierung von DNS-Include-Dateien

Das im folgenden abgedruckte Skript erzeugt die */var/named/**-Tabellen, welche über "$include"-Statements in die Haupttabellen des Nameservers (hier Bind 8.2.X bzw. 9.X.Y) eingelesen werden. Es wird am besten mit den Benutzerrechten des Nameserverdaemons (**named**) aufgerufen.

```
#!/usr/bin/perl
```

```
#
# Skript zur Generierung von DNS-Includes aus der
# Rechner-/HardWare-DB
#
# Dieses Skript dient als Arbeitsvorlage, da viele Spezifika der
# Testumgebung noch "fest verdrahtet" sind.
#
# Dirk von Suchodoletz <dirk@goe.net>, 01-03-2001
#
#################################################################

# Subroutinen zur Typumwandlung
sub btodq {
        my $dq = join ".",unpack("CCCC",$_[0]);
        return $dq;
}
sub dqtob {
        my @dq;
        my $q;
        my $i;
        my $bin;

        foreach $q (split /\./,$_[0]) {
                push @dq,$q;
        }
        for ($i = 0; $i < 4 ; $i++) {
                if (! defined $dq[$i]) {
                        push @dq,0;
                }
        }
        $bin    = pack("CCCC",@dq);
        return $bin;
}

# Bestimmen der IP-Adressen des DNS-Servers

$ipcfg = `/sbin/ifconfig` ;
@ipcfg = split  /^/, $ipcfg ;

$k=0;
foreach $i (@ipcfg) {
$_ = $i ;
if ( /inet/ ) {
  @i = split ;
  @j = split ( /\:/, $i[1] ) ;
   $ip[$k] = $j[1] ;
  @j = split ( /\:/, $i[2] ) ;
   $broadc[$k] = $j[1] ;
  @j = split ( /\:/, $i[3] ) ;
   $netm[$k++] = $j[1] ; }
}
```

```
# Verbindung zur Datenbank
use Mysql;
$DB = Mysql->connect(s8, Hardwareverwaltung, hwvdb);

# Formatdefinition

format OUTHost =
@<<<<<<<<<<< @<<  @<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$HostName  'IN' $val1  $val2
.

# Q&D Hack fuer DNS-Reverse
open (OUTRev1, ">/var/named/134.76.60.rev.inc");
open (OUTRev2, ">/var/named/134.76.61.rev.inc");
open (OUTRev3, ">/var/named/10.9.rev.inc");

# Abfrageschleife
$k=0;
while ($ip[$k]) {
    print "Generiere DNS-DB fuer Server-IP: $ip[$k]\n";
$sth01=$DB->query("select IPAddress, HostName,
  DomainName, Grouping.Name FROM Grouping, Rechner WHERE
  FirstDNS='$ip[$k]' OR SecondDNS='$ip[$k]' AND
  Rechner.GroupingID=Grouping.GroupingID
  ORDER BY DomainName,HostName;");
$i=1;
while ( @data = $sth01->fetchrow ) {
   $DomainName[$i] = $data[2];
   $IPAddress = $data[0];
   $HostName = $data[1];
   if ( $data[2] ne $DomainName[$i-1] ) {
 $DomainName[$i] = $data[2];
 close (OUTHost);
# Wechseln der Ausgabedatei und Schreiben eines Headers mit
# Datum etc.
 open (OUTHost, ">/var/named/$DomainName[$i].db.inc");
 ($sec,$min,$hour,$mday,$mon,$year,
$wday,$yday,$isday) = localtime (time);
 print OUTHost ";\n;";
 print OUTHost "DNS-Include generated for Domain: ";
 print OUTHost $DomainName[$i++],"\n;\n; --> ";
 print OUTHost $year+1900,"/",$mon+1,"/",$mday+1;
         print OUTHost " $hour:$min";
         print OUTHost "\n;\n; (c) Dirk von Suchodoletz";
 print OUTHost " <dirk\@goe.net>, 2001\n;\n";
   }
$val1 = "A";
$val2 = $IPAddress;
# Formatierte Ausgabe 'A'-Record
write OUTHost;
# Formatierte Ausgabe des TXT-Feldes
$HostName = "";
```

```
$val1 = "TXT";
$val2 = '"'.$data[3].'"';
write OUTHost;
# Nachsehen, ob Aliases fuer eine bestimmte IP definiert sind
$sth02=$DB->query("SELECT Value FROM Rechner,
  Rechner_Properties,PropertyNamen WHERE
  Rechner.RechnerID=Rechner_Properties.RechnerID AND
  Rechner_Properties.PropertyNameID=39 AND
  Rechner_Properties.PropertyNameID=
   PropertyNamen.PropertyNameID
  AND Rechner.IPAddress='$IPAddress';");
# Aliases bei Bedarf formatiert ausgeben
while ( @data02 = $sth02->fetchrow ) {
   $val1 = "CNAME";
   $val2 = $data[1];
   $HostName = $data02[0];
   write OUTHost;
}
# Schreiben der Reverse-Dateien (Q&D-Hack)
$_ = $IPAddress;
if (/134\.76\.60/) {
   @ipn = split (/\./) ;
   print OUTRev1 "$ipn[3]\tIN\tPTR\t$data[1]\n"; }
if (/134\.76\.61/) {
   @ipn = split (/\./) ;
   print OUTRev2 "$ipn[3]\tIN\tPTR\t$data[1]\n"; }
if (/10\.9\./) {
   @ipn = split (/\./) ;
   print OUTRev3 "$ipn[2].$ipn[3]\tIN\tPTR\t$data[1]\n"; }

}

$k++;
}
```

Anhang E

Inventarisierungstools

E.1 Reports

Mittels des Perl-Skriptes **report** lassen sich aus der Datenbank für einzelne Rechner bzw. Rechnergruppen Bestandsaufnahmen erheben. Zum einen werden die für das einzelne Gerät registrierten Hardwarekomponenten angezeigt und zum anderen alle wichtigen Netzwerkparameter aufgeführt.
report wird an der Kommandozeile aufgerufen und nimmt als Parameter die Option "-ps" für die Ausgabe eines Postscriptdokuments für die Inventur der Einzelteile an. Einen Beispielausdruck findet man auf Seite 135 des III. Teils. Anschließend können einzelne Rechnernamen oder durch Leerzeichen getrennt Listen von Rechnernamen angegeben werden. Das Ergebnis wird in die Datei *report.txt* für die Konfigurations- und Teileliste bzw. *report.ps* für die Postscriptdatei für die Inventurerfassung im aktuellen Verzeichnis gespeichert. Fehlermeldungen und Warnhinweise, die z.B. auf eine unvollständige Hardwareausstattung hinweisen, finden sich in *report.log*. Auch diese Meldung erfolgt im aktuellen Verzeichnis, wobei in allen Fällen eine Schreibberechtigung Voraussetzung ist.
Die Postscriptvorlagendatei *psheader.ps* wird im Verzeichnis */usr/share/adm* erwartet. Sie wird für den Textausdruck nicht benötigt. Zu Beginn der Datei sind die Zugriffsparameter für die Datenbank vermerkt; entsprechende Kommandozeilenoptionen könnten dieses generalisieren.

```
#!/usr/bin/perl
#
# Skript zur Generierung von Reports aus der Rechner/
# HardWare-DB, Version 0.2.0b
#
# Dirk von Suchodoletz <dirk@goe.net>, 19-06-2001
#
```

```
#####################################################################
sub line {
my $OUT = shift;
    print $OUT "---------------------------------";
print $OUT "---------------------------------\n";
}


$i = 0;
foreach ( @ARGV ) {
if (/[^-ps]/) { $rechner[$i++]=$_; }
if (/^-ps/) { $PS = 'y'; }
}


# Verbindung zur Datenbank
use Mysql;
$DB = Mysql->connect(s15, Hardwareverwaltung, hwvdb);

# Zieldatei oeffnen
if ($PS eq 'y') {
# PostScriptdatei erzeugen
open (OUT, ">report.ps");
open (IN, "/usr/share/adm/psheader.ps");
print OUT "%!ps\n\%output produced by program report\n";
print OUT "\%(c) Internet-AG 2001\n";
print OUT "\%(c) Dirk von Suchodoletz <dirk\@goe.net>,";
print OUT " (19-06-2001)\n%\n";
while (<IN>) {
print OUT $_; }
close (IN);
}
# Textdatei erzeugen
else {
open (OUTD, ">report.txt");
# Formatdefinition fuer Textdatei
format OUTD =
@<<<<<<<<<<<<<<<@>>>>>>>>>>>>>>>
@<<<<<<<<<<<<<<<@>>>>>>>>>>>>>>>
$val1 $val2 $val3 $val4
.
format OUTH =
@<<<<<<<<<<<<<<<<<<<<<<<<<  @<<<<<<<<<<<  @<<<<<<<<<<
@>>>>>>>
$val10 $val20 $val30 $val40
   @<<<<<<<<<<<<<<<<<<<<<<<<  @<<<<<<<<<<<@<<<<<<<<<<
    @<<<<<<<<<
$val11 $val21 $val31 $val41
.
}
# In jedem Fall Errorlog schreiben
open (ERR, ">report.log");
```

```
($sec,$min,$hour,$mday,$mon,$year,
  $wday,$yday,$isday) = localtime (time);

#################################################################

$i = 0;
foreach ( @rechner ) {
    $sth01=$DB->query("SELECT * FROM Rechner WHERE
HostName='$rechner[$i]';");
while ( @data = $sth01->fetchrow ) {
$RechnerID = $data[0];
if ( $data[1] ne '') { $IPAddress = $data[1]; }
else { $IPAddress = "Not set"; }
if ( $data[2] ne '') { $HostName = $data[2]; }
else { $HostName = "Not set"; }
if ( $data[3] ne '') { $GroupingID = $data[3]; }
else { $GroupingID = "Not set"; }
if ($PS ne 'y') {
open (OUTD, ">>report.txt");
print OUTD "#\n# Report generated for RechnerID";
print OUTD " $RechnerID\n#\n# --> ",$year+1900,"-";
print OUTD $mon+1,"-",$mday," $hour:$min";
print OUTD "\n#\n# (c) Dirk von Suchodoletz ";
print OUTD "<dirk\@goe.net>, 2001\n#\n\n";
print OUTD "Basisdaten des Rechners:\n";
line(OUTD);
$val1 = "Hostname:";
$val2 = $HostName;
$val3 = "IP-Adresse:";
$val4 = $IPAddress;
write OUTD;
$val1 = "Gruppe:";
$val2 = $GroupingID;
$val3 = $val4 ='';
write OUTD;
}
# Anzeigen der Gruppendaten
if ( $data[3] ne '' && $data[3] ne '0') {
$sth02=$DB->query("SELECT * FROM Grouping WHERE
GroupingID='$GroupingID';");
@data02 = $sth02->fetchrow;
$GroupName = $data02[1];
$NetMask = $data02[2];
$DomainName = $data02[3];
$Gateway = $data02[4];
$FirstDNS = $data02[5];
$SecondDNS = $data02[6];
$Broadcast = $data02[7];
if ($PS eq 'y') {
open (OUT, ">>report.ps");
# IP-Adresse fuer Strichcode (EAN) umwandeln
```

```
            @IPwoD = split(/\./,$IPAddress) ;
            $IPwoD = sprintf "%3d%3d%3d%3d", $IPwoD[0],
$IPwoD[1], $IPwoD[2], $IPwoD[3];
            $IPwoD =~ s/\s/0/g ;
print OUT "/x_txt 60 def\n/Times-Bold 45 (Rechner: ";
print OUT $RechnerID,") 775 0";
print OUT " zeilel\n/x_txt 574 def\n/Times-Bold  45 ";
print OUT "($HostName) 775 0 zeiler\n";
print OUT "/x_txt 300 def\n/Times-Bold 72";
print OUT "(Inventurliste) 710 0.80 zeilek\n";
print OUT "/x_txt 70 def\n/Times-Roman 9 ";
print OUT "(Internet-AG - Platz der Goettinger Sieben";
print OUT " 5 - 37073 Goettingen) 681 0 zeilel\n";
print OUT "/x_txt 532 def\n/Times-Roman 14 ($mday.";
print OUT $mon+1,".",$year+1900,") 674 0 zeiler\n";
print OUT "/Times-Roman 14 (Internet-AG) 652 0 zeiler\n";
print OUT "/Times-Roman 14 (Universitaet Goettingen)";
print OUT "636 0 zeiler\n/Times-Roman 14 ";
print OUT "(Platz der Goettinger Sieben 5) 618 0\n";
print OUT " zeiler /Times-Roman 14 (Telefon: 39-8392)";
print OUT "600 0 zeiler\n";
print OUT "($HostName)\n($DomainName)\n($IPAddress)\n";
print OUT "(Internet-AG)\n($mday.",$mon+1,".";
print OUT $year+1900,")\n(<hotline\@stud.uni-goettingen";
print OUT ".de>)\n(Inventarliste Hardware)\n";
print OUT "($RechnerID)\n($IPwoD)\n";
print OUT "32 510\nrkleber\n\n";
print OUT "/x_txt 574 def\n";
print OUT "/Times-Roman 8 ((c) 2001 Dirk von\n";
print OUT " Suchodoletz <dirk\@goe.net>) 15 0 zeiler\n";
}
else {
print OUTD "\nDie Gruppe $data02[0] hat folgende ";
print OUTD "Eigenschaften (Basisdaten):\n";
line(OUTD);
$val1 = "Gruppenname:";
$val2 = $GroupName;
$val3 = "NetMask:";
$val4 = $NetMask;
write OUTD;
$val1 = "Broadcast:";
$val2 = $Broadcast;
$val3 = "GateWay:";
$val4 = $Gateway;
write OUTD;
print OUTD "DomainName:\t",$DomainName,"\n";
$val1 = "Erster DNS-Server:";
$val2 = $FirstDNS;
$val3 = "Zweiter DNS-Server:";
$val4 = $SecondDNS;
line (OUTD);
}
```

```
# weitere, variable Daten abfragen
$sth02=$DB->query("SELECT Name, Value FROM
Grouping_Properties, PropertyNamen WHERE
PropertyNamen.PropertyNameID=
Grouping_Properties.PropertyNameID
AND GroupingID='$GroupingID' ORDER BY
PropertyNamen.PropertyArtID, Name;");
$j=0;
while ( @data = $sth02->fetchrow ) {
    if ( $j++ % 2 == 0 ) {
$val1 = $data[0].":";
$val2 = $data[1]; }
else {
$val3 = $data[0].":";
$val4 = $data[1];
write OUTD; }
}
if ( $j % 2 == 0) {
    print $data[0], ": ", $data[1], "\n"; }
}
# Vermerk, falls keine Gruppe eingetragen ist
else {
   print ERR "\nDer Rechner $HostName ist fuer keine";
   print ERR "Gruppe vermerkt\n";
}
close (OUTD);
print OUTH "\nEingesetzte HardWare:\n";
$sth03=$DB->query("SELECT HardWareArten.Art,HardWareNamen.Name,
  Beschafft, Garantie_bis, BruttoPreis, StrichCode FROM
  HardWare, HardWareNamen, HardWareArten WHERE
  HardWareArten.HardWareArtID=HardWareNamen.HardWareArtID
  AND HardWare.HardWareNameID=HardWareNamen.HardWareNameID
  AND HardWare.RechnerID=$RechnerID ORDER BY Art;");
if ($PS eq 'y') {
$j=0;
while ( @data03 = $sth03->fetchrow ) {
$field[$j][0] = $data03[0];
$field[$j][1] = $data03[1];
$field[$j][2] = $data03[2];
$field[$j][3] = $data03[3];
$field[$j][4] = $data03[4];
$field[$j][5] = $data03[5];
$j++;
}
print OUT "/x_txt 60 def\n";
print OUT "/Times-Bold 12 (Produktname) 560 0 zeilel\n";
$k=0;
while ($k<$j) {
print OUT "/Times-Roman 10 (",$field[$k++][1],") ";
print OUT 556-$k*28," 0 zeilel\n"; }
print OUT "/x_txt 78 def\n";
print OUT "/Times-Bold 12 ((Produktart)) 544 0 zeilel\n";
```

```
$k=0;
while ($k<$j) {
print OUT "/Times-Roman 10 ((",$field[$k++][0],")) ";
print OUT 544-$k*28," 0 zeilel\n"; }
print OUT "/x_txt 222 def\n";
print OUT "/Times-Bold 12 (Beschafft) 544 0 zeilek\n";
$k=0;
while ($k<$j) {
print OUT "/Times-Roman 10 (",$field[$k++][2],") ";
print OUT 544-$k*28," 0 zeilek\n"; }
print OUT "/x_txt 290 def\n";
print OUT "/Times-Bold 12 (Garantie_bis) 544 0 zeilek\n";
$k=0;
while ($k<$j) {
print OUT "/Times-Roman 10 (",$field[$k++][3],") ";
print OUT 544-$k*28," 0 zeilek\n"; }
print OUT "/x_txt 325 def\n";
print OUT "/Times-Bold 12 (Seriennummer) 560 0 zeilel\n";
print OUT "/x_txt 380 def\n";
$k=0;
while ($k<$j) {
print OUT "/Times-Roman 10 (",$field[$k++][5],") ";
print OUT 550-$k*28," 0 zeiler\n"; }
print OUT "/x_txt 345 def\n";
print OUT "/Times-Bold 12 ((intern)) 544 0 zeilel\n";
print OUT "/x_txt 480 def\n";
print OUT "/Times-Bold 12 (Brutto-) 560 0 zeilel\n";
print OUT "/x_txt 496 def\n";
print OUT "/Times-Bold 12 (Preis) 544 0 zeilel\n";
print OUT "/x_txt 526 def\n";
$k=$GesPr=0;
while ($k<$j) {
print OUT "/Times-Roman 10 (",$field[$k][4]," DM) ";
print OUT 522-$k*28," 0 zeiler\n";
$GesPr+=$field[$k++][4];}
$GesPr = sprintf "%.2f", $GesPr;
print OUT "/Times-Bold 10 (",$GesPr," DM) ";
print OUT 522-$k*28," 0 zeiler\n";
print OUT "/x_txt 454 def\n/Times-Bold 12 ";
print OUT "(Gesamtpreis:)",522-$k*28," 0 zeiler\n";
print OUT "/Times-Bold 12 (Barcode) 552 0 zeiler\n";
$k=0;
print OUT "395 540 translate\n";
while ($k<$j) {
print OUT "0 ",-28," translate\n";
print OUT "(",$field[$k++][5],") barcode\n"; }
print OUT " showpage\n";

}
else {
open (OUTH, ">>report.txt");
line(OUTH);
```

```
$val10 = "Produktname";
$val11 = "(Hardwareart)";
$val20 = "Beschafft";
$val21 = "Garantie";
$val30 = "Strichcode";
$val31 = "(intern)";
$val40 = "EPreis";
$val41 = "(brutto)";
write OUTH;
line(OUTH);
while ( @data03 = $sth03->fetchrow ) {
# print "$data03[1]\t$data03[0]\t$data03[5]\n"; }
$val10 = $data03[1];
    $val11 = '('.$data03[0].')';
$val20 = $data03[2];
$val21 = $data03[3];
$val30 = '  '.$data03[5];
$val31 = '';
$val40 = $data03[4];
$GesPr += $data03[4];
$val41 = '';
write OUTH; }
line(OUTH);
$val11 = $val20 = $val21 = $val31 = $val41 = '';
$val10 = "Gesamtpreis:";
$val30 = "incl. MwST.";
$val40 = sprintf "%.2f", $GesPr;
write OUTH;
}
}
$i++;
$GesPr = 0;

line(OUTH);
close (OUTH);
}
close (OUT);
close (ERR);
```

E.2 Generieren von Aufklebern

Zur äußeren Kennzeichnung der im Netzwerk betriebenen Rechner und zur Identifizierung der einzelnen Hardwarekomponenten lassen sich mit diesem Beispielwerkzueg Aufkleber für "Zweckform"-Bögen[1] generieren. Das Perl-Skript **label** übernimmt diese Aufgabe. Steht ein spezieller Labeldrucker zur Verfügung sind entsprechende Anpassungen des Ausgabeformates notwen-

[1] Üblicherweise Din-A4 zum Ausdruck über Standarddrucker

dig. Die Postskriptvorlagen können der beigelegten CD entnommen werden, da diese zum Abdruck zu umfangreich sind.

```
#!/usr/bin/perl
#
# Skript zur Generierung von Aufklebern aus der Rechner/
# HardWare-DB, Version 0.8.0a
#
# Dirk von Suchodoletz, <dirk@goe.net> 16-06-2001
#
################################################################
#
$i = $j = 0;
foreach ( @ARGV ) {
# Monitoretikett
    if (/^-m/) {
if ( $etk[$i][0] eq "Inventarlabel") {
die ("Fehler:
 Nur ein Etikettentyp auf einmal druckbar!\n") }
else {
    $etk[++$i][0] = "Monitor_Etikett";
    print "Arg M erk.\n";
    $me = 'y'; $j = 0; }
}
# Rechneretikett
elsif (/^-r/) {
if ( $etk[$i][0] eq "Inventarlabel") {
    die ("Fehler:
 Nur ein Etikettentyp auf einmal druckbar!\n") }
else {
    $etk[++$i][0] = "Rechner_Etikett";
    print "Arg R erk.\n"; $re = 'y'; $j = 0; }
}
# Offset
elsif (/^-o/) {
$offs = 'y'; $me = $re = 'n'; }
# Verantwortlich
elsif (/^-v/) {
$verantwortlich = "Internet-AG 2001"; }
# Hardwareetikett (Teil, einfach bedruckt)
elsif (/^-t$/) {
if ( $etk[$i][0] eq "Monitor_Etikett" ||
    $etk[$i][0] eq "Rechner_Etikett") {
    die ("Fehler:
 Nur ein Etikettentyp auf einmal druckbar!\n")
}
else {
    $etk[0][0] = 'tkleber';
    $tl = 'y';
}
}
# Hardwareetikett (Teil, doppelt bedruckt)
```

```
elsif (/^-tt$/) {
if ( $etk[$i][0] eq "Monitor_Etikett" ||
    $etk[$i][0] eq "Rechner_Etikett") {
    die ("Fehler:
 Nur ein Etikettentyp auf einmal druckbar!\n")
}
else {
    $etk[0][0] = 'ttkleber';
    $tl = 'y';
}
}


# Ist kein Optionsschalter, also Wert ...

# Rechnername einlesen
elsif (/[^-]/ && ($re eq 'y' || $me eq 'y')) {
$etk[$i][++$j]=$_; }

elsif (/[^-]/ && $offs eq 'y' && /[0-9]{1,2}/) {
$offset = $_; $offs = 'n'; }

elsif (/[^-]/ && $verantwortlich eq 'Internet-AG 2001')
{ $verantwortlich = $_; }

elsif (/[^-]/ && $tl eq 'y') {
if (/^[0-9]{5}$/ && $start eq '' &&
    $start[0] eq '') { $start = $_; }
elsif (/^[0-9]{5}$/) {
    if (!$start[0]) { $ende = $_; }
    else { $start[$j++] = $_; }
}
elsif (/^[0-9]{5}\,$/) {
    $start[$j++] = $_;
    print "Einzele.\n";}
elsif (/^[0-9]{6}\,$/) {
    $start[$j++] = $_;
    $nop = 'y'; }
elsif (/^[0-9]{6}$/) {
    $start[$j++] = $_;
    $nop = 'y'; }
else {
    die ("Fehler:
 Nur Numerische Inventarlabel zulaessig bzw. Zahl
 zu kurz/lang!\n");
}
}
}
#################################################################

# Plausibilitaetsueberpruefungen der Eingabe
if ( $start ne '' && $ende eq '' ) {
```

```
die ("Endmarke nicht gesetzt!\n"); }

# Defaults setzen
if ( $verantwortlich eq '' ) {
$verantwortlich = 'Internet-AG 2001'; }

print "Offset: $offset\nStart: $start\nEnde: $ende\n";

################################################################

# Template (fuer alle Typen einheitlich) einlesen und in
# Zieldatei schreiben
open (IN, "/usr/share/adm/psheader.ps");
open (OUT, ">label.ps");
print OUT "%!ps\n\%output produced by program label\n";
print OUT "\%(c) Internet-AG 2001\n";
print OUT "\%(c) Dirk von Suchodoletz <dirk\@goe.net>,";
print OUT " (13-03-2001)\n%\n";
while (<IN>) {
print OUT $_; }
close (IN);

################################################################

# Unterscheidung nach zu druckenden Aufklebern

# Sollen Etikettreihen generiert werden
if ($tl && $start && !$start[0]) {
# Bestimmung des Offsets
$x = (150*($offset%4));
$y = (780-60*(int($offset/4)));

for ($i = $start ; $i <= $ende ; $i++) {
if ($x == 600) {
$x = 0;
$y -= 60; }
# Abbruchbedingung: Unterer Seitenrand kann nicht kleiner Null
# sein
if ( $y < 0 ) { last; }
# Wenn Pruefziffer nicht gegeben -> berechnen, sonst einfach
# Input ausgeben
if (!$nop) {
@code = split (//,$start);
# Der Originalstring wird zur Pruefsummenberechnung zerlegt und
# die Laufvariable in Abhangigkeit ihrer Laenge aufaddiert
if ($i-$start < 10) {
$code[4] += $i-$start; }
else {
@icode = split (//,$i-$start);
print "I-Code: $icode[0], $icode[1] und $icode[2]\n";

$code[3] += $icode[0];
```

```
$code[4] += $icode[1];
}
# Pruefsummenberechnung
$code[5] = sprintf("%.0f",(($code[0]*20+$code[1]*2
+$code[2]*4+$code[3]*6+$code[4]*5)/30));
# print "Position: $x, $y, $code[5]\n";
# Ausgabe fuer PS-Ausdruck
if ($etk[0][0] eq 'ttkleber') {
print OUT "($code[0]$code[1]$code[2]$code[3]$code[4]";
print OUT "$code[5])\n$x $y\n$etk[0][0]\n\n"; }
else {
print OUT "($code[0]$code[1]$code[2]$code[3]$code[4]";
print OUT "$code[5])\n($verantwortlich)\n";
print OUT "$x $y\n$etk[0][0]\n\n"; }
}
else {
$_ =~ s/,//g ;
if ($etk[0][0] eq 'ttkleber') {
print OUT "($_)\n$x $y\n$etk[0][0]\n\n"; }
else {
print OUT "($_)\n($verantwortlich)\n";
print OUT "$x $y\n$etk[0][0]\n\n"; }
}
# Naechste Ausgabe eine Position nach links verschieben
$x += 150;
}
}
# Sollen vordefinierte Teilelabel gedruckt werden ...
elsif ($tl && !$start && $start[0]) {
# Bestimmung des Offsets
$x = (150*($offset%4));
$y = (780-60*(int($offset/4)));

foreach (@start) {
if ($x == 600) {
$x = 0;
$y -= 60; }
# Abbruchbedingung: Unterer Seitenrand kann nicht kleiner Null
# sein
if ( $y < 0 ) { last; }
# Wenn Pruefziffer nicht gegeben -> berechnen, sonst einfach Input
# ausgeben
if (!$nop) {
@code = split (//,$_);
# Pruefsummenberechnung
$code[5] = sprintf("%.0f",(($code[0]*20+$code[1]*2
+$code[2]*4+$code[3]*6+$code[4]*5)/30));
# print "Position: $x, $y, $code[5]\n";
# Ausgabe fuer PS-Ausdruck
if ($etk[0][0] eq 'ttkleber') {
print OUT "($code[0]$code[1]$code[2]$code[3]$code[4]";
print OUT "$code[5])\n$x $y\n$etk[0][0]\n\n"; }
```

```
else {
print OUT "($code[0]$code[1]$code[2]$code[3]$code[4]";
print OUT "$code[5])\n($verantwortlich)\n";
print OUT "$x $y\n$etk[0][0]\n\n";
}
}
else {
$_ =~ s/,//g ;
if ($etk[0][0] eq 'ttkleber') {
print OUT "($_)\n$x $y\n$etk[0][0]\n\n"; }
else {
print OUT "($_)\n($verantwortlich)\n";
print OUT "$x $y\n$etk[0][0]\n\n"; }
}
# Naechste Ausgabe eine Position nach links verschieben
$x += 150;
}
}
# Monitor- oder Rechneraufkleber
else {
# Verbindung zur Datenbank
use Mysql;
$DB = Mysql->connect(s15, Hardwareverwaltung, hwvdb);

# Aktuelle Systemzeit ermitteln und formatieren
($sec,$min,$hour,$mday,$mon,$year,
  $wday,$yday,$isday) = localtime (time);
$mon += 1; $year += 1900;
# Bestimmung des Offsets
$x = (280*($offset%2));
$y = (600-120*(int($offset/2)));

$i = 0;
while ( $etk[++$i] ) {
print "Schleife drin\n";
$etikett = $etk[$i][0];
$j=0;
while ( $etk[$i][++$j] ) {
print "Bla: $etk[$i][$j]\n";
if ($x >= 560) {
$x = 0;
$y -= 120; }
# Abbruchbedingung: Unterer Seitenrand kann nicht kleiner Null
# sein
if ( $y < 0 ) { last; }

$rechner = $etk[$i][$j];

$sth01=$DB->query("select * from Rechner where
HostName='$rechner';");
while ( @data = $sth01->fetchrow ) {
$RechnerID = $data[0];
```

```
if ( $data[1] ne '') { $IPAddress = $data[1]; }
else { die ("IP-Adresse nicht bekannt\n"); }
$HostName = $data[2];
if ( $data[3] ne '') { $GroupingID = $data[3]; }
else { die ("Rechner keiner Gruppe zugrordnet\n"); }
if ( $data[3] ne '' && $data[3] ne '0') {
    $sth02=$DB->query("select DomainName, Art from
Grouping, RechnerArten where
Grouping.RechnerArtID=RechnerArten.RechnerArtID
AND GroupingID='$GroupingID';");
@data02 = $sth02->fetchrow;
     $DomainName = $data02[0];
$RechnerArt = $data02[1];
}
# IP-Adresse fuer Strichcode (EAN) umwandeln
@IPwoD = split(/\./,$IPAddress) ;
$IPwoD = sprintf "%3d%3d%3d%3d", $IPwoD[0], $IPwoD[1],
$IPwoD[2], $IPwoD[3];
$IPwoD =~ s/\s/0/g ;
print OUT "($HostName)\n($DomainName)\n($IPAddress)\n";
print OUT "($verantwortlich)\n($mday.$mon.$year)\n";
print OUT "(\\(c\\) by Dirk von Suchodoletz ";
print OUT "<dirk\@goe.net>)\n";
# Etikettauswahl fuer Rechnertyp 'Diskless X-Station'
if ($RechnerArt eq 'Diskless X-Station') {
if ($etikett eq 'Monitor_Etikett') {
print OUT "(Bitte 'localhost' nehmen, ";
print OUT "da sonst USB)\n";
print OUT "(Sound, Floppy, ... nicht lokal ";
print OUT "erreichbar sind!)\n";
print OUT "(D   X   S)\n"; }
elsif ($etikett eq 'Rechner_Etikett') {
print OUT "(Bitte 'localhost' nehmen!)\n";
print OUT "(D  X  S)\n";
print OUT "($IPwoD)\n"; }
}
# Etikettauswahl fuer Rechnertyp 'X-Terminal'
elsif ($RechnerArt eq 'X-Terminal') {
if ($etikett eq 'Monitor_Etikett') {
print OUT "(Bei Systemstillstand Strg-Alt";
print OUT "-Backspace)\n(bzw. anschliessend ";
print OUT "die Reset-Taste druecken)\n";
print OUT "(X - Terminal)\n"; }
elsif ($etikett eq 'Rechner_Etikett') {
print OUT "(Bei Problemen->Tel:(39-)8392)\n";
print OUT "(Terminal)\n";
print OUT "($IPwoD)\n"; }
}
# Etikettauswahl fuer Rechnertyp 'X-Terminal-Server'
elsif ($RechnerArt eq 'X-Terminal-Server') {
if ($etikett eq 'Monitor_Etikett') {
print OUT "(Diesen Rechner NIE zwischen)\n";
```

```
print OUT "(08:00 - 22:00 Uhr ausschalten!!)\n";
print OUT "(X - Server)\n"; }
elsif ($etikett eq 'Rechner_Etikett') {
print OUT "(Rechner NIE ausschalten!)\n";
print OUT "(X-Server)\n";
print OUT "($IPwoD)\n"; }
}
# Etikettauswahl fuer Rechnertyp 'Server'
elsif ($RechnerArt eq 'Server') {
if ($etikett eq 'Monitor_Etikett') {
print OUT "(Diesen Rechner NIE ausschalten,)\n";
print OUT "(sondern sauber herunterfahren!!)\n";
print OUT "(Server)\n"; }
elsif ($etikett eq 'Rechner_Etikett') {
print OUT "(Rechner NIE ausschalten!)\n";
print OUT "(Server)\n";
print OUT "($IPwoD)\n"; }
}
# Etikettauswahl fuer restliche Rechnertypen
else {
if ($etikett eq 'Monitor_Etikett') {
print OUT "(Rechner bitte sauber mittels)\n";
print OUT "(Strg-Alt-Del herunterfahren!)\n";
print OUT "(Linux Workstation)\n"; }
elsif ($etikett eq 'Rechner_Etikett') {
print OUT "(Bei Problemen->Tel:(39-)8392)\n";
print OUT "(Workst.)\n";
print OUT "($IPwoD)\n"; }
}
# Positionierung
print OUT "$x $y\n";
if ($etikett eq 'Rechner_Etikett') {
print OUT "rkleber\n\n"; }
else {
print OUT "mkleber\n\n"; }
# Druckposition nach links verschieben
$x += 280;
  }
}
}
}
#################################################################

# Ausgabe der generierten Seite aller Etiketttypen

print OUT "\n\nshowpage\n";
close (OUT);
```

Zur Generierung von Rechneraufklebern (mit oder ohne Strichcode) greift das Programm auf die Datenbank zu, so dass nur der Name des Rechners angegeben werden muss. Ein Datenbankzugriff braucht für Teile-Label nicht zu erfolgen, könnte jedoch in einer erweiterten Version vorgesehen werden,

Option	Syntaxbeispiel	Bedeutung
-m	-m rechner01 [rechner02 ...]	Erzeugen eines Aufklebers ohne Strichcode
-r	-r rechner01 [rechner02 ...]	Erzeugen eines Aufklebers mit Strichcode
-o	-o INT	Verschieben der Startpostion des ersten zu druckenden Aufklebers
-t	-t 12345	Erzeugen eines Teile-Labels, wobei die Prüfzahl automatisch generiert wird
-t	-t 123456	Erzeugen eines Teile-Labels mit bereits gegebener Prüfzahl
-tt	-tt 12345	Erzeugt Teile-Label (wie -t) jedoch doppelt pro Aufkleber
-v	-v Verantwortliche(r) / Eigentümer	Ändert den Standardeintrag auf andere Person bzw. Organisation

Tabelle E.1: Kommandozeilenoptionen von **label**

um festzustellen, ob ein bestimmtes Label bereits in der Datenbank registriert ist. So kann verhindert werden, dass beim Nachdruck Kennzeichnungen doppelt vergeben werden. Zu Beginn des Skriptes sind die Informationen zum Zugriff auf die Datenbank festgehalten, dieses könnte jedoch auch über entsprechende Kommandozeilenschalter geschehen. Die Sicherheitsimplikationen sind an dieser Stelle jedoch nicht besonders kritisch, wenn ein ReadOnly-Zugriff für diese Kategorie von Werkzeugen eingerichtet wurde. Werden hier strengere Maßstäbe angelegt, wird man um eine interaktive Passwortabfrage oder schärfere Authentifizierungsmechanismen nicht herumkommen.
Normale selbstklebende Papieraufkleber genügen den meisten Anforderungen. Steht nicht ein spezieller Drucker für solche Label zur Verfügung, arbeitet man am besten mit klassischen A4-Bögen[2], welche über einen normalen Drucker laufen können. Das entsprechende Ausgabeprogramm ist an die Gegebenheiten anzupassen.

[2]Im beschriebenen Problem handelt es sich dabei um Zweckformbögen.

Anhang F

Administrationstools

F.1 Generierung der *exports*-Datei

Der Gerätetyp Diskless X-Station erlaubt das Arbeiten des Benutzers direkt auf der Maschine. Hierzu werden nicht nur die entsprechenden Verzeichnisse für das Starten von Programmen gemountet werden müssen, sondern auch die Homeverzeichnisse der jeweils angemeldeten Benutzer. Damit entsteht der Bedarf nach einer stets aktuellen */etc/exports* Datei zur Steuerung des Verhaltens des NFS-Servers, welcher die Homeverzeichnisse verwaltet.

```
#!/usr/bin/perl
#
# Skript zur Generierung von Reports aus der Rechner/
# HardWare-DB, Version 0.2.1a
#
# Dirk von Suchodoletz <dirk@goe.net>, 21-05-2001
#
##############################################################

# Verbindung zur Datenbank
use Mysql;
$DB = Mysql->connect(s15, Hardwareverwaltung, hwvdb);

# Zieldatei oeffnen
open (OUT, ">exports");

($sec,$min,$hour,$mday,$mon,$year,
  $wday,$yday,$isday) = localtime (time);

print OUT "
# /etc/exports
#
# Automatically generated by mk_exports.pl\n#\n#";
```

```
print OUT " --> ",$year+1900,"/",$mon+1,"/",$mday+1;
print OUT " $hour:$min";
print OUT "
#
# (c) Dirk von Suchodoletz <dirk\@goe.net>, 2001\n#\n";

################################################################

@dir=("/home", "/home/users", "/home/special", "/home/user0",
 "/home/user1", "/home/user2", "/home/user3", "/home/user4");

$sth01=$DB->query("SELECT HostName from Rechner, Grouping
 where Rechner.GroupingID=Grouping.GroupingID AND
 Grouping.RechnerArtID != 2 ORDER BY RechnerArtID,HostName;");
$i=0;
while ( @data = $sth01->fetchrow ) {
$rechner[$i++]=$data[0]; }

foreach ( @dir ) {
  print OUT $_,"\n\t-root=s1:ag4-02.gwdg.de:jupiter\n";
  print OUT "\t-rw=$rechner[0]";

  for ($j=1; $j < $i ; $j++) {
   print OUT ":",$rechner[$j]; }
  print OUT "\n\t-access=$rechner[0]";
  for ($j=1; $j < $i ; $j++) {
   print OUT ":",$rechner[$j]; }
  print OUT "\n";
}
```

Das hier vorgestellte Tool **mk_exports** generiert anhand der Datenbank die Liste der Freigaben für ein Dec-Unix-system. Entsprechende Anpassungen für andere Unixes richten sich nach der Syntax der jeweiligen *exports* Datei.

F.2 Installation und Update

F.2.1 Serviceboot-Diskette

Diese Bootdiskette enthält ein minimales DOS-System, am besten FreeDOS, da so keine Lizenzprobleme entstehen. Inzwischen liegt ein vorbereitetes Diskettenimage der SuSE-Distribution bei, da dieses auch für andere Konfigurationsaufgaben, wie die Einstellung von Netzwerkkarten mit ihren DOS-Utilities benötigt wird. Auf diese Weise kann sichergestellt werden, dass die Servicediskette sowohl innerhalb einer Linuxumgebung als auch unter den verschiedenen Microsoft-Betriebssystemen einsetzbar ist. Auf die-

ser Diskette befinden sich neben den Systemdateien, eine *autoexec.bat*[1] in welcher *loadlin.exe* mit der Parameterdatei *params.lin* aufgerufen wird:

```
vmlinuz nfsroot=134.76.10.19:/nfsroot/dxs \
    nfsaddrs=134.76.10.31:134.76.10.19    \
    :134.76.10.254:255.255.255.0:hostname:eth0:none
```

Die Parameterdatei konfiguriert analog zu Etherboot den Kernel mit den Basis-IP-Daten[2] und der NFS-Root-Umgebung[3]. Dem "Bootprompt- HOWTO" kann man das Aussehen der Parameterzeile entnommen werden. Weiterhin liegt auf der Diskette der aktuelle Kernel der DXS bevor mit **mknbilinux**[4] er "getagged" wurde (**vmlinuz**). Dieses vermeidet die Abhängigkeit von der Erreichbarkeit eines DHCP-Servers und kann bei entsprechender Einrichtung der NFS-Freigaben in der */etc/exports* des Servers in einem weitgefaßten Netzwerkbereich diese Servicediskette einsetzen. Der Startvorgang erfolgt nach dem Laden des Kernels analog zu den DXS, wobei das Fehlschlagen der umfassenden Konfiguration per **dhclient** in den meisten Fällen keine für die anstehenden Aufgaben negativen Auswirkungen hat[5].
Alle zur Installation oder Wartung eines Systems notwendigen Module werden bereits durch den Betrieb der DXS vorliegen bzw. müssen im Zuge der Erstellung des DXS-Kernels mit vorgesehen werden. Diese werden dann bei Bedarf entweder mittels des Kernel-Modulladers, welcher per */etc/ modules.conf* eingestellt wird, automatisch geladen oder vom Administrator per **insmod** oder **modprobe** eingefügt. Bei entsprechender Defaultkonfiguration wird der SSH-Daemon gestartet, so dass die Maschine auch von wenig versiertem Personal gestartet und anschließend fernadministriert oder -installiert werden kann. Unabhängig von dem Wunsch Linux auf einer solchen Maschine zu installieren, können auf dieser Basis viele Systeminformationen zur Hardwareausstattung[6] beschafft werden und Tests zur Überprüfung der Funktionsfähigkeit der Maschine laufen. Die Voraussetzung bildet in beiden Fällen natürlich eine geeignete Netzwerkanbindung.

[1] Meistens genügt hier der Eintrag der Zeile `loadlin @params.lin`

[2] nfsaddr=Client IP:Server IP:Gateway:Netmask: Hostname:Interface:Option

[3] nfsroot= Server IP:/NFS-Pfad

[4] Siehe hierzu den Abschnitt B.4

[5] Die Grafische Oberfläche wird meistens nicht benötigt und Informationen zu Druck-, Fontservern ... sind nicht zwingend erforderlich. Möchte man dieses eleganter gestalten, könnte der Aufruf der Zusatzkonfiguration von der Kernelversion o.ä. abhängig gemacht werden.

[6] Die Kernelschnittstelle des *proc*-Filesystems liefert Informationen zur CPU, PCI-Devices, Ressourcenzuweisung (Interrupts, IO-Adressen, DMA-Ports), weitere Programme können das PnP-Interface auslesen und darstellen. Die Kernelbootmeldungen zeigen gefundene und initialisierte Hardware...

F.2.2 Das "autoconfig"-Skript

Nicht alle Rechner werden als DXS oder X-Terminal zu betreiben sein. Mindestens eine Reihe von Servern und Rechner mit Spezialaufgaben (DNS-, Web-, Mailserver) werden über Festplatten verfügen müssen. Auch für diese Systeme ist es jedoch sinnvoll eine einheitliche Grundinstallation vorzunehmen, die möglichst wenig von anderen Maschinen abweicht, um den Wartungsaufwand zu verringern.
Die Autoinstallationsumgebung basiert auf dem Dateisystem der Diskless X-Stations, wobei alle notwendigen Dateien im Beispiel unterhalb von */var/adm/config* liegen. Notwendige Kernelmodule für spezielle Aufgaben (Raidkonfiguration, LVM[7], ...) sind entsprechend beim Erstellen des Kernels für die Diskless X-Stations mit vorzusehen.
Beim Durchführen der Installation wird von der Bootdiskette automatisch ein Backup angelegt, welches später wieder zum Erstellen evtl. Ersatz- oder Servicedisketten für diese Maschine zur Verfügung steht.
autoconfig kennt zwei Kommandozeilenoptionen: "-bd Devicename" gibt das Bootdevice und "-rd Devicename" gibt das Rootdevice für die Installation an. Alle weiteren für die Installation notwendigen Daten werden aus der Datenbank beschafft, deren Zugriffskontrolle über entsprechende Einträge des Tools erfolgt. Bis auf die Vorbereitung des Dateisystems mit Partitionieren und Formatieren, auf das aus Sicherheitsgründen[8], läuft die Installation anschließend automatisch ab.

```
#! /usr/bin/perl -w
#
# Skript zur "automatischen" Systeminstallation
#
# Die Installation wird von einem "Master-Server" abgeleitet
# und an die lokalen Gegebenheiten angepasst. Dieses Skript
# ist als Beispiel zu verstehen und implementiert laengst
# nicht alle Moeglichkeiten.
#
# Dirk von Suchodoletz <dirk@goe.net>, 08-09-2001
#
$version = "0.2.1e" ;
#
#################################################################
# Subroutinen zur Typumwandlung
sub btodq {
        my $dq = join ".",unpack("CCCC",$_[0]);
```

[7] Logical Volume Manager, welcher es erlaubt mehrere Festplatten(partitionen) zu einer einzigen zusammenzufassen. RAID, das Redundant Array of Inexpensive Disks bietet diese Funktionalität schon länger, erlaubt aber kein dynamisches Entfernen und Hinzufügen von Geräten.

[8] Vermeiden des Löschens wichtiger Daten

```
        return $dq;
}
sub dqtob {
        my @dq;
        my $q;
        my $i;
        my $bin;
        foreach $q (split /\./,$_[0]) { push @dq,$q; }
        for ($i = 0; $i < 4 ; $i++) {
                if (! defined $dq[$i]) { push @dq,0; }
        }
        $bin    = pack("CCCC",@dq);
        return $bin;
}

#####################################################################
# Einlesen von Kommandozeilenoptionen

foreach ( @ARGV ) {
    if (/^-rd/ && !$rd) { $rd = "0"; }
    elsif (/^-bd/ && !$bd) { $bd = "0"; }
    elsif (/\/[a-z0-9\/]{2,}/) {
if ($rd eq "0") { $rd = $_; }
elsif ( $bd eq "0" ) { $bd = $_; }
    }
}

print "Kontrolle: Bootdev: $bd Rootdev: $rd\n";

print "Die Systeminstallation basierend auf einer SuSE 7.X ";
print "wird gestartet mit autoconfig der Version: $version.\n" ;


# determine nfs server. we assume that database and rsync
# server are identical to the nfs. if not change it below

print "Determine MAC address of the network adaptor and";
print " gather other\ndata about network configuration.\n";

# Ermittele aus /proc/mount den NFS-Server

open (LOG, ">/tmp/install.log") ||
die "Kann Logfile nicht oeffnen\n";
open (IN, "/proc/mounts") ||
die "/proc/mounts ist nicht zu oeffen\n";

foreach $i (<IN>) {
$_ = $i ;
        if ( /nfs/ ){
@i = split ( /addr\=/ )  ;
@i = split ( / /, $i[1] ) ;
$nfsserver = $i[0] ; }
```

```
}
close (IN) ;

# Setze den Datenbank und Rsync-Server

$dbserver=$nfsserver;
$rsyncserver=$nfsserver;

# Ermittele die MAC-Adresse

$ipcfg = `/sbin/ifconfig` ;
@ipcfg = split  /^/, $ipcfg ;

foreach $i (@ipcfg) {
$_ = $i ;

if ( /HWaddr/ ) {

  @i = split ;
  $macaddress=$i[4] ; }
}
#####################################################################
# Verbinde zur Datenbank (alle Parameter sind statisch einge-
# tragen, koennten aber spaeter auch ueber Kommandozeilenpara-
# meter gesetzt werden

use Mysql;

$dbh = Mysql->connect("10.16.10.35","Hardwareverwaltung","hwvdb");

# Ermittele die Basis Netzwerkparameter aus der Datenbank

$sth = $dbh->query("select HostName, DomainName, IPAddress,
 NetMask, BroadCast, GateWay, FirstDNS, SecondDNS,
 Grouping.GroupingID, Rechner.RechnerID, RechnerArtID
 from HardWare, HardWare_Properties, Rechner, Grouping where
 Rechner.RechnerID=HardWare.RechnerID AND
 HardWare_Properties.Value='$macaddress' AND
 HardWare.HardWareID=HardWare_Properties.HardWareID
 AND Grouping.GroupingID=Rechner.GroupingID;");

@config=$sth->fetchrow ;

$hostname = $config[0] ;
$domain_name = $config[1] ;
$ipaddress = $config[2] ;
$netmask = $config[3] ;
$broadcast = $config[4] ;
if ($config[5] ne "") {
$router = $config[5] ; }
else {
    $router = $config[2] ; }
```

```
$domain_name_servers = $config[6] ;
if ($config[7] ne "") {
$domain_name_servers .= ",".$config[7]; }
$groupid = $config[8];
$computerid = $config[9];
$sysartid = $config[10];

################################################################
# Setze einige Standardeintraege fuer bestimmte Variablen

$displaymng='console';
$start_rwho=$start_x11='no';
$start_snmp='no';
$start_atalk=$start_named='no';
$start_httpd=$start_smb='no';
$start_dnetc='yes';
$ntalkd="/usr/sbin/in.ntalkd" ;
$otalkd="/usr/sbin/in.talkd" ;
$axstart="no" ;

# Ermittele weitere Netzwerkparameter aus der Gruppeninformation

$sth03=$dbh->query("SELECT Grouping_Properties.Value,
    PropertyNamen.Name FROM Grouping_Properties,
    PropertyNamen WHERE
    Grouping_Properties.GroupingID='$groupid' AND
    Grouping_Properties.PropertyNameID=
PropertyNamen.PropertyNameID");

while ( @add_data = $sth03->fetchrow )
{
 SWITCH: {
  ($add_data[1] eq 'XDM') && do { $displaymng=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'RWHO') && do { $start_rwho=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'CRON') && do { $start_cron=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'X11') && do { $start_x11=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'SNMP') && do { $start_snmp=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'DNETC') && do { $start_dnetc=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'SMB') && do { $start_smb=$add_data[0];
          last SWITCH; };
  ($add_data[1] eq 'HTTPD') && do
{ $start_httpd=$add_data[0]; last SWITCH; };
  ($add_data[1] eq 'NIS-Domain') && do
{ $ypdomain=$add_data[0]; last SWITCH; };
  ($add_data[1] eq 'Mail-Server') && do
{ $mailserver=$add_data[0]; last SWITCH; };
```

```
  ($add_data[1] eq 'News-Server') && do
{ $newsserver=$add_data[0]; last SWITCH; };
  ($add_data[1] eq 'Proxy-Server') && do
{ $proxyserver=$add_data[0]; last SWITCH; };
  ($add_data[1] eq 'IRC-Server') && do
{ $ircserver=$add_data[0]; last SWITCH; };
  ($add_data[1] eq 'X-Login-Server') && do
{ if ( $x_display_manager eq '' )
     { $x_display_manager=$add_data[0]; }
          else
        { $x_display_manager.=' '.$add_data[0]; }
  last SWITCH; };
  ($add_data[1] eq 'NTP-Server') && do
        { if ( $ntp_servers eq '' )
{ $ntp_servers=$add_data[0]; $start_xntpd='yes'; }
          else { $ntp_servers.=', '.$add_data[0]; }
          last SWITCH; };
  ($add_data[1] eq 'Log-Server') && do {
          if ( $log_servers eq '' )
{ $log_servers=$add_data[0]; }
          else { $log_servers.=', '.$add_data[0]; }
          last SWITCH; };
  ($add_data[1] eq 'LPR-Server') && do {
          if (!$lpr_servers) { $lpr_servers=$add_data[0]; }
          else { $lpr_servers.=', '.$add_data[0]; }
          last SWITCH; };
  ($add_data[1] eq 'Font-Server') && do {
          if (!$font_servers) { $font_servers=$add_data[0]; }
          else { $font_servers.=', '.$add_data[0]; }
          last SWITCH; };
  ($add_data[1] eq 'DomainSearch') && do {
          if (!$dnssearch) { $dnssearch=$add_data[0]; }
          else { $dnssearch.=', '.$add_data[0]; }
          last SWITCH; };
  ($add_data[1] eq 'NIS-Server') && do {
          if (!$ypserv) { $ypserv=$add_data[0]; }
          else { $ypserv.=', '.$add_data[0]; }
          last SWITCH; };
     }
}

#################################################################
# Ermittele nun notwendige Hardware-Daten

# Bestimme das Mausprotokoll
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE
    HardWare.RechnerID=$computerid AND
    HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    HardWareNamen.HardWareArtID=2 AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID AND
```

```
    HardWareNamen_Properties.HardWareNamen_PropertyNameID=56");
    if ( @add_data = $sth2->fetchrow ) {
$mp = $add_data[0]; }
    else {
print LOG "Crit: Setze fuer $hostname Maus-Protokoll auf ";
        print LOG "PS/2 da nicht in Datenbank\n";
        $mp = "PS/2"; }

# Ermittele den Mausanschlusstyp
$sth2=$dbh->query("SELECT PropertyNamen.Name FROM
    HardWareNamen_Properties, HardWareNamen, HardWare,
PropertyNamen WHERE PropertyNamen.PropertyArtID=3 AND
    PropertyNamen.PropertyNameID=
    HardWareNamen_Properties.HardWareNamen_PropertyNameID AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen.HardWareArtID=2 AND
HardWare.RechnerID=$computerid
    AND HardWare.HardWareNameID=HardWareNamen.HardWareNameID");
    if ( @add_data = $sth2->fetchrow ) {
$_ = $add_data[0];
     if (/Serial/) { $md = "ttyS0"; }
     else { $md = "psaux"; } }
    else {
        print LOG "Crit: Setze fuer $hostname Maus-Anschluss";
        print LOG " auf psaux da nicht in Datenbank\n";
        $md = "psaux"; }

# Ermittele Tastatursprache
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE HardWare.RechnerID=$computerid
AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    HardWareNamen.HardWareArtID=3 AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=53");
    if ( @add_data = $sth2->fetchrow ) {
$key_lang = $add_data[0]; }
    else {
print LOG "Warn: Setze fuer $hostname Tastatur-Default-";
print LOG "Sprache auf Deutsch\n";
$key_lang = "de"; }

# Ermittele die Monitordaten
# 1) Max. Bildschirmaufloesung welche konfiguriert wird
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE HardWare.RechnerID=$computerid
AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    HardWareNamen.HardWareArtID=1 AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=2");
```

```
    if ( @add_data = $sth2->fetchrow ) {
$max_res = $add_data[0]; }
    else {
print LOG "Warn: Setze fuer $hostname Default-Monitor-";
print LOG "Aufloesung auf 1024x768 da nicht in Datenbank\n";
$max_res = "1024x768"; }

# 2) Max. Vertikalfreq.
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE HardWare.RechnerID=$computerid
AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    HardWareNamen.HardWareArtID=1 AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=13");
    if ( @add_data = $sth2->fetchrow ) {
$vfreq = $add_data[0]; }
    else {
print LOG "Crit: Nehme fuer $hostname Monitor Vertikalfreq.";
print LOG "von 65kHz da nicht in Datenbank\n";
$vfreq = "65"; }

# 3) Max. Horizontalfreq.
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE HardWare.RechnerID=$computerid
AND HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    HardWareNamen.HardWareArtID=1 AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=14");
    if ( @add_data = $sth2->fetchrow ) {
$hfreq = $add_data[0]; }
    else {
print LOG "Crit: Nehme fuer $hostname Monitor Horizontalfreq.";
print LOG "von 90Hz da nicht in Datenbank\n";
$hfreq = "90"; }

# 5) XFree86-Treiber (Version 4.0.X)
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE
HardWare.RechnerID=$computerid AND
    HardWareNamen.HardWareNameID=HardWare.HardWareNameID AND
    (HardWareNamen.HardWareArtID=5 OR
    HardWareNamen.HardWareArtID= 23) AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID
    AND HardWareNamen_Properties.HardWareNamen_PropertyNameID=10");
if ( @add_data = $sth2->fetchrow ) {
$xdriver = $add_data[0]; }
else {
print LOG "Warn: Setze fuer $hostname XFree86-Modul auf";
print LOG "'vga' (generic)\n";
```

```
$xdriver = 'vga'; }

# 6) Umfang des Graphik-RAM
$sth2=$dbh->query("SELECT Value FROM HardWare, HardWareNamen,
    HardWareNamen_Properties WHERE HardWare.RechnerID=
$computerid AND HardWareNamen.HardWareNameID=
HardWare.HardWareNameID AND
    (HardWareNamen.HardWareArtID=5 OR
HardWareNamen.HardWareArtID=23) AND
    HardWareNamen_Properties.HardWareNameID=
HardWareNamen.HardWareNameID AND
    HardWareNamen_Properties.HardWareNamen_PropertyNameID=7");
if ( @add_data = $sth2->fetchrow ) {
$graphicram = $add_data[0]; }
else {
print LOG "Warn: Setze fuer $hostname GraphicRamgroesse";
print LOG "auf 2048kByte\n";
$graphicram = 2048; }

# Ermittele mit /etc/init.d/boot.local nachzuladende Module
$sth2=$dbh->query("SELECT Value FROM HardWare,
HardWareNamen_Properties WHERE HardWare.RechnerID=
'$computerid' AND
    HardWareNamen_Properties.HardWareNameID=
HardWare.HardWareNameID AND
HardWareNamen_Properties.HardWareNamen_PropertyNameID='15'");

while ( @add_data = $sth2->fetchrow ) {
   if ( $boot_local eq '' ) {
$boot_local='modprobe -qa '.$add_data[0]; }
   else {
$boot_local.=' '.$add_data[0]; }
   }
if ( $boot_local ne '' ) {
$boot_local.=' &>/dev/null'; }

#####################################################################
# Starte nun den Installationsprozess

# Erzeuge alle notwendigen Systemverzeichnisse; einige werden
# nicht mittels "rsync" angelegt, da sie vom Kopieren
# ausgeschlossen werden (z.B. /var, /tmp ...)

print "Erzeuge alle notwendigen Verzeichniseintraege ...\n" ;

open (IN,"directories") ;
while (<IN>) {
chomp ;
mkdir $_, 0755 ; }
close (IN) ;

# Erzeuge leere utmp und wtmp
```

```
open (OUT, ">/mnt/var/run/utmp") ;
print OUT "" ;
close (OUT) ;
open (OUT, ">/mnt/var/log/wtmp") ;
print OUT "" ;
close (OUT) ;
open (OUT, ">/mnt/var/adm/config/config.newinst") ;
print OUT "" ;
close (OUT) ;
print "Filesystemvorbereitungen sind abgeschlossen.\n" ;

# Erzeuge ein Backup der computerspezifischen Bootdiskette

print "\nBackup der Bootdiskette ...\n" ;
system("dd if=/dev/fd0 of=/mnt/var/adm/config/bootdisk.backup&");

# Merke den Rsyncserver fuer die Verwendung im "Update"-Skript

open (OUT, ">/mnt/var/adm/config/rsync-server") ||
    die "Kann die Datei nicht anlegen!\n" ;
print OUT $rsyncserver,"\n" ;
close (OUT) ;

# Konfiguriere Font-Server

print "\nKonfiguriere Fontserver ...\n" ;
open (OUT, ">/mnt/etc/FONTSERVER") ||
    die "Can't open /etc/FONTSERVER\n" ;
$fs = $font_servers;
#$fs =~ /,/\ /;
print OUT $fs;
close OUT ;

open (OUT, ">/mnt/var/adm/config/inst-software") ||
    die "Can't open file \n" ;

#################################################################
# Starte den Kopiervorgang mittels des Rsync-Prozesses

use Expect ;

print "Starte den Rsync-Prozess zum Kopieren des Filesystems.\n";

# Es wird ein Passwort benoetigt, wenn der Rsync-Server
# geschuetzt sein soll.

$passwd = "bla" ;

($control = Expect->spawn("rsync -a -z -e ssh
--include-from=rsync.include --exclude-from=rsync.exclude
rsync:\/\/rsync\@$nfsserver\/Whole-System\/ \/mnt")) ;
unless ($control->expect(30,"Password:")) {
```

```
die "Das Passwort ist falsch\n" ; }
print $control "$passwd\r" ;
unless ($control->expect(3000,"")) { ; }

print "Der Systemabgleich ist abgeschlossen ...\n" ;

#####################################################################
# Konfiguration der meisten Systemvariablen (meist in /etc)
# Erzeuge einige fehlende Variablen

print "Erzeuge fehlende Netzwerkvariablen ...\n" ;

$netname = btodq ( dqtob( $ipaddress ) & dqtob( $netmask ) );

if (!$newsserver) { $newsserver="news"."."$domain_name ; }
if (!$ircserver) { $ircserver="irc"."."$domain_name ; }

# Definiere YP Konfiguration, falls erforderlich

if ($ypdomain ne "" && $ypserv ne "") {
  $ypconf="yes" ;
  $startypb="yes";}
else {
  $ypconf="no" ;
  $startypb="no";}

$xntps=$ntp_servers;
$xntps=~s/,// ;

$domain_name_servers=~s/^\s+// ;
$domain_name_servers=~s/\s+$// ;

$_=$domain_name_servers ;
@dns=split (/,/) ;

$dns1=$dns[0]." ".$dns[1]." ".$dns[2] ;

$dns1=~s/^\s+// ;
$dns1=~s/\s+$// ;

$dnssearch =~ s/^\s+// ;
$dnssearch =~ s/\s+$// ;

$_=$dnssearch ;
@dnssearch=split (/,/) ;
$dnssearch1=$dnssearch[0]." ".$dnssearch[1]." ".$dnssearch[2];

$dnssearch1=~s/^\s+//;
$dnssearch1=~s/\s+$//;

for ($i=0; $i<3; $i+=1) {
```

```
if ($dnssearch[$i] ne "") {
$fw[$i] = "" ;}
else  { $fw[$i] = "//" ;}

if ($dns[$i] ne "") {
$sl[$i] = "" ;}
else  { $sl[$i] = "//" ;}
}

$reverse="in-addr.arpa" ;

for ($i=0; $i<4; $i+=1) {
 if ((oct $netname[$i]) != 0) {
  $reverse = (oct $netname[$i]).".".$reverse;}
}

#####################################################################
# Erzeuge einige Konfigurationsdateien direkt
# Generiere die /etc/fstab

@fdisk = `/sbin/fdisk -l` ;

system("cp etc.tmpl/fstab.tmpl /mnt/etc/fstab") ;

open (OUT, ">>/mnt/etc/fstab") || die "Can't open fstab\n" ;

print OUT "$rd\t\/\t\tauto    defaults\t   1  1\n" ;

# Versuche den Swapspeicher zu ermitteln und dann einzutragen
foreach $i (@fdisk) {
$_ = $i ;
        if ( /swap/ ){
@i = split ;
system("mkswap $i[0]") ;
print OUT $i[0],"\tswap\t\tswap    defaults,pri=1\t  0  0\n" ;
    }
}
close (OUT);


#####################################################################
# Die folgende Liste bestimmt die zu bearbeitenden Konfi-
# gurationsdateien

@netconfig_dateien=( "snmpd.conf", "named.conf",
 "webalizer.conf", "exim.conf", "issue", "issue.net",
 "exports", "xinetd.conf", "lilo.conf", "hosts",
 "rc.config", "dhcpd.conf", "route.conf", "smb.conf",
 "proftpd.conf", "httpd/httpd.conf", "mtools.conf",
 "resolv.conf", "inittab", "init.d/boot.local") ;
foreach $i (@netconfig_dateien) {
```

```
  open (IN, "etc.tmpl/$i.tmpl") ||
die "Can't open etc.tmpl/$i.tmpl\n" ;
  open (OUT, ">/mnt/etc/$i") || die "Can't open /etc/$i \n" ;

# Die Platzhalter werden durch ihre entsprechenden Variablen
# ersetzt

 while (<IN>) {
s/AC-VERSION/$version/o ;
s/IP-ADDR/$ipaddress/o ;
s/IP-BROAD/$broadcast/o ;
s/IP-NETMASK/$netmask/o ;
s/HOST-NAME/$hostname/go ;
s/DOMAIN-NAME/$domain_name/o ;
s/NET-NAME/$netname/o ;
s/D-N-S/$domain_name_servers/o ;
s/FORW1/$dns[0]/o ;
s/FW1/$fw[0]/o ;
s/FORW2/$dns[1]/o ;
s/FW2/$fw[1]/o ;
s/REVERSE/$reverse/o ;
s/SLAVE1/$dnssearch[0]/o ;
s/SL1/$sl[0]/o ;
s/SLAVE2/$dnssearch[1]/o ;
s/SL2/$sl[1]/o ;
s/SLAVE3/$dnssearch[2]/o ;
s/SL3/$sl[2]/o ;
s/DNS-SEARCH/$dnssearch/o ;
s/DNSSEARCH/$dnssearch1/o ;
s/DNS/$dns1/o ;
s/ROUTER/$router/o ;
s/MOUSEDEV/$md/o ;
s/NAMESERVER=/NAMESERVER\=\"$dns1\"/o ;
s/DISPLAYMNG/$displaymng/o ;
s/BOOTDEVICE/$bd/o ;
s/N-TALKD/$ntalkd/o ;
s/O-TALKD/$otalkd/o ;
s/X-CONNECT/$start_x11/o ;
s/^KEYTABLE.*$/KEYTABLE=\"$key_lang\"/o ;
s/START_NAME.*$/START_NAMED="$start_named"/o ;
s/START_ATALK.*$/START_ATALK="$start_atalk"/o ;
s/NNTPSERV.*$/NNTPSERVER="$newsserver"/o ;
s/IRCSERV.*$/IRCSERVER="$ircserver"/o ;
s/CREATE_YP.*$/CREATE_YP_CONF="$ypconf"/o ;
s/YP-SERVER/$ypserv/o ;
s/YP-DOMNAME/$ypdomain/o ;
s/START_YPBIND.*$/START_YPBIND="$startypb"/o ;
s/NFS_SERVER=.*$/NFS_SERVER="yes"/o ;
s/START_HTT.*$/START_HTTPD="$start_httpd"/o ;
s/START_SMB.*$/START_SMB="$start_smb"/o ;
s/START_XNTPD.*$/START_XNTPD="$start_xntpd"/o ;
s/XNTPD_INITIAL.*$/XNTPD_INITIAL_NTPDATE="$xntps"/o ;
```

```
s/START_SNMP.*$/START_SNMPD="$start_snmp"/o ;
s/START_RWHOD.*$/START_RWHOD="$start_rwho"/o ;
s/START_MYSQ.*$/START_MYSQL="no"/o ;
s/CRON.*$/CRON="$start_cron"/o ;
s/START_DNETC.*$/START_DNETC="$start_dnetc"/o ;
s/START_DHCP.*$/START_DHCPD="yes"/o ;
s/START_EXIM.*$/START_EXIM="yes"/o ;
        s/START_AXNET.*$/START_AXNET="$axstart"/o ;
s/GPM_PARAM.*$/GPM_PARAM=" -t $mp -m \/dev\/$md"/o ;
s/PROXYSERV/$proxyserver/o ;
s/ROOTDEVICE/$rd/o ;
s/BOOTDEVICE/$bd/o ;
s/BOOTLOCAL/$boot_local/o ;
print OUT $_ ;
  }
  close (IN) ;
  close (OUT) ;
}
print "... done \n";

#####################################################################

# Erzeuge nun die /etc/X11/XF86Config

system("touch /mnt/var/log/xdm.errors") ;

$cdp = 16;

open (IN, "etc.tmpl/XF86Config.tmpl") ;
open (OUT, ">/mnt/etc/X11/XF86Config") ||
    die "Can't open XF86Config.Start\n" ;

@maxres = ("640x480", "800x600", "1024x768", "1280x1024",
"1600x1200");
foreach (@maxres) {
$modes = "\"$_\" \"lcd$_\" ".$modes;
if ($_ eq $max_res) { last; }
}

while(<IN>) {
s/AC-VERSION/$version/ ;
s/LANG/$key_lang/ ;
s/RAM/$graphicram/ ;
s/MP/$mp/ ;
s/MD/$md/ ;
s/VS/$vfreq/ ;
s/HS/$hfreq/ ;
s/DRV/$xdriver/ ;
s/CDP/$cdp/ ;
s/MODES/$modes/ ;
print OUT $_ ;
}
```

```
close (IN) ;
close (OUT) ;

# Erzeuge /etc/X11/xdm/Xaccess

open (OUT ,">/mnt/etc/X11/xdm/Xaccess");
print OUT "*\n%hostlist\t$hostname $x_display_manager\n";
print OUT "*\t\tCHOOSER \%hostlist\n";
close (OUT);

#####################################################################

print "Konfiguriere den Kernel und trage das Root-Device ein\n" ;

system("cp -a boot.tmpl/* /mnt/boot");
system("/usr/sbin/rdev /mnt/boot/vmlinuz $rd") ;
system("/usr/sbin/rdev /mnt/boot/vmlinuz.orig $rd") ;
print "Installing LiLo ...\n" ;
system("lilo -C /etc/lilo.conf -r /mnt");

#####################################################################
# Kopiere einige spezielle Konfigurationsdateien

print "Kopiere spezielle Konfigurationsdateien ...\n" ;
system("cp etc.tmpl/raidtab etc.tmpl/aliases etc.tmpl/*.orig
etc.tmpl/motd etc.tmpl/passwd etc.tmpl/shadow /mnt/etc") ;
system("ln -sf /opt/kde2/bin/chooser /mnt/etc/X11/xdm/chooser");
system("cp -a bin/cron.daily* /mnt/root/bin");
system("cp -a etc.tmpl/exclude_local etc.tmpl/motd.config
/mnt/var/adm/config");

# Erledige einige Spezialanpassungen, die sich schlecht
# generalisieren lassen (mittels Zusatzskripten)

if ( $sysartid==3 || $sysartid==5 ) {
  print "Fuege spezielle Anpassungen hinzu ...\n";
  system("spec/config.studiserv"); }
elsif ( $sysartid==4 ) {
  print "Fuege spezielle Anpassungen hinzu ...\n";
  system("spec/config.hotline"); }

print "Autoconfig wurde abgeschlossen!\n" ;

#####################################################################
```

F.2.3 Das Programm "update"

Dieses Tool erlaubt den Abgleich des lokalen Filesystems gegen das eines (Master-)Servers. Das Skript basiert auf der interpretierten Sprache Perl und verwendet für die Netzwerktransfers RSYNC. Der **rsyncd** wird als Daemon entweder permanent oder über den **(x)inetd** auf dem Server ge-

startet und mittels der Datei */etc/rsyncd.conf* gesteuert. Evtl. vergebene Passwörter werden in der Datei */etc/rsyncd.secrets*, welche nur vom Systemadministrator gelesen werden darf, hinterlegt.
Das Verhalten des Clients **rsync** wird mittels Exclude- und Includelisten gesteuert, welche u.a. im Verzeichnis */var/adm/config* neben der Datei, welche die IP-Adresse des Rsync-Servers (*rsync-server*) enthält, liegt. Mit der Kommandozeilenoption "–nobackup" kann das Anlegen eines Backups der evtl. zu löschenden Dateien verhindert werden, was den Austauschvorgang in einigen Fällen stark beschleunigen kann. Die meisten Exclude-Listen (*rsync.**) werden vom Server übernommen und orientieren sich an dessen Software-Ausstattung und ihren Erfordernissen. Möchte man verhindern, dass bestimmte Dateien und Verzeichnisse beim Update berücksichtigt werden, müssen diese in der Datei *exclude_ local* hinterlegt werden. Generierte Backups werden im Verzeichnis */var/adm/backup* in komprimiertem Format hinterlegt. Während des Updatevorganges muss dann jedoch ausreichend Festplattenplatz zur Verfügung stehen. Gezeigt wird im Folgenden nur **update.pl**, ein Wrapper-Skript **update** kann vorgeschaltet werden, um auch ein Update der Skripten selbst sauber zu erreichen.

```
#! /usr/bin/perl -w
#
# (SuSE) linux auto update version 0.0.4
#
# Dieses Skript wird von einem Wrapper (update) aufgerufen,
# welcher wiederum das Update des Skripts selbst erlaubt.
#
# Dirk von Suchodoletz <dirk@goe.net>, 26-04-2001
#
################################################################

$nobackup="no";

foreach ( @ARGV ) {
if (/nobackup/) { $nobackup="yes"; }
}

$passwd = "bla" ;


print "starting all update \n" ;

################################################################
# software parameters

$rsyncserver=`cat /var/adm/config/rsync-server` ;
chomp $rsyncserver ;

print $rsyncserver, ": Rsyncserver ...\n" ;
```

```
#############################################################
# rsync system

use Expect ;

print "starting system main rsync ...\n" ;

$passwd = "bla" ;
if ( $nobackup ne "yes" ) {
($control = Expect->spawn("rsync -a -z -v -n --delete
-e ssh --exclude-from=exclude_local
--exclude-from=rsync.exclude --include-from=rsync.include
rsync://rsync\@$rsyncserver/Whole-System/ / \|
grep \"deleting\" | grep -v \" directory \" \|
awk \'\{print \"tar -rlf /var/adm/backup/system.backup
/\"\$2 \}\'\> backup.system")) ;
unless ($control->expect(30,"Password:")) {
die "Password is incorrect \n" ; }
print $control "$passwd\r" ;
unless ($control->expect(3000,"")) { ; }
chmod 0700, "backup.system" ;
system ("./backup.system") ;
system ("bzip2 -f /var/adm/backup/system.backup &") ;
# system ("rm backup.system") ;
}
($control = Expect->spawn("rsync -a -z -v --delete -e ssh
--exclude-from=exclude_local
--exclude-from=rsync.exclude
--include-from=rsync.include
rsync://rsync\@$rsyncserver/Whole-System/ / ")) ;
unless ($control->expect(30,"Password:")) {
die "Password is incorrect \n" ; }
   print $control "$passwd\r" ;
   unless ($control->expect(3000,"")) { ; }

system("cp update.new update");
system("ldconfig");

print "System rsync complete ...\n" ;
```

F.3 Funktionsüberwachung

F.3.1 Netzwerkmonitoring mittels "tkined"

Das Programm **tkined**[9] stellt eine plattformübergreifende Möglichkeit der Funktionsüberwachung einzelner Netzwerkkomponenten zur Verfügung. Dieses umfaßt einfache Ping-Tests zur Überprüfung der Erreichbarkeit bis hin zum Monitoring einzelner SNMP-Variablen.

[9] siehe hierzu [W7]

Aus der Datenbank läßt sich mittels **mk_tkined** die Konfigurationsdatei für dieses Programm generieren. Nach dem Aufruf des Programms stehen im aktuellen Verzeichnis drei Dateien zur Verfügung: *server.tki*, die Konfigurationsdatei zur Überwachung der Serverfunktionen, *dxs.tki* zur Überprüfung der Arbeitsstationen, die direkte Benutzeranmeldung erlauben, der DXS und *xterminal.tki* zur Überwachung der X-Terminals. Hierbei werden jeweils unterschiedliche Variablen überwacht, die Last der Netzwerkinterfaces, der Auslastungsgrad des Dateisystems, die Zahl der Benutzer, die Load der Maschine und die Temperatur der CPU's bei den Servern sowie die Netzwerklast, Innentemperatur der Thin-Clients. Andere Variablen sind bei Bedarf denkbar, erfordern jedoch entsprechende Module des SNMP und die Fähigkeit des Tools zur Darstellung.
Voraussetzung für die Zusammenarbeit mit **mk_tkined** ist die aktuellste Version 1.5 von **tkined**. Dieses basiert auf der aktuellen TCL/TK-Installation, die auch auf anderen Unixes lauffähig sein sollte.

```
#!/usr/bin/perl
#
# Skript zur Generierung der TKIned-Konfiguration aus der
# Rechner/HardWare-DB, Version 0.7.0
#
# Dirk von Suchodoletz, <dirk@goe.net> 25-04-2001
#
################################################################

sub stripchart {
my $sc_nr=shift;
my $x=shift;
my $y=shift;
my $hn=shift;
my $ip=shift;
my $sc_str=shift;
print OUT "set stripchart$sc_nr [ ined -noupdate create";
print OUT " STRIPCHART ]\nined ";
print OUT "-noupdate move \$stripchart$sc_nr $x.00 $y.00\n";
print OUT "ined -noupdate font \$stripchart$sc_nr fixed\n";
print OUT "ined -noupdate color \$stripchart$sc_nr Black\n";
print OUT "ined -noupdate scale \$stripchart$sc_nr 100.00\n";
print OUT "ined -noupdate size \$stripchart$sc_nr\n";
print OUT "ined -noupdate name \$stripchart$sc_nr $hn\n";
print OUT "ined -noupdate address \$stripchart$sc_nr $ip\n";
print OUT "ined -noupdate attribute \$stripchart$sc_nr ";
print OUT "SNMP:Config {-address ";
print OUT $ip," -port 161 -version SNMPv1 -community ";
print OUT "public -transport udp -timeout 2 -retries 2";
print OUT " -window 10 -delay 2 -tags {}}\n",$sc_str,"\n";
}

sub group {
```

```
my $gn=shift;
my $x=shift;
my $y=shift;
my $lb=shift;
my $ge=shift;
my $icon=shift;
print OUT "set group$gn [ ined -noupdate create GROUP";
print OUT " $ge ]\nined ";
print OUT "-noupdate move \$group$gn -9999999 -9999999\n";
print OUT "ined -noupdate move \$group$gn $x $y";
print OUT "\nined -noupdate icon \$group$gn $icon\nined ";
print OUT "-noupdate font \$group$gn fixed\n";
print OUT "ined -noupdate color \$group";
print OUT "$gn Black\nined -noupdate name \$group$gn {";
print OUT "$lb}\nined -noupdate label \$group$gn name\n\n";
}

sub node {
my $nr=shift;
my $x=shift;
my $y=shift;
my $hn=shift;
my $ip=shift;
my $icon=shift;
print OUT "set node$nr [ ined -noupdate create NODE ]\n";
print OUT "ined -noupdate move \$node$nr $x $y\n";
print OUT "ined -noupdate icon \$node$nr $icon\n";
print OUT "ined -noupdate font \$node$nr fixed\n";
print OUT "ined -noupdate color \$node$nr Black\n";
print OUT "ined -noupdate name \$node$nr ", $hn;
print OUT "\nined -noupdate address \$node$nr ", $ip;
print OUT "\nined -noupdate attribute \$node$nr ";
print OUT "Monitor:ThresholdAction flash\nined ";
print OUT "-noupdate attribute \$node$nr {round trip time}";
print OUT "{rtt 1 ms}\nined -noupdate attribute \$node$nr ";
print OUT "Monitor:RisingThreshold 20\nined -noupdate";
print OUT " label \$node$nr name\n\n";
}

sub header {
my $fh=shift;
open (OUT, ">$fh");

print OUT "#!/bin/sh\n##";
print OUT " This file was created by tkined version 1.5.0";
print OUT "   >> DO NOT EDIT <<\n##\n## This may look like ";
print OUT "TCL code but it is definitely not! \\\n";
print OUT "exec tkined \"\$0\" \"\$@\"\n\nined page A4";
print OUT " landscape\n\n# (c) Internet-AG 2001\n";
print OUT "# (c) Dirk von Suchodoletz <dirk\@goe.net>,";
print OUT " (13-03-2001)\n#\n";
print OUT "# Automatically generated by dhcp-generate.pl\n#";
```

```
($sec,$min,$hour,$mday,$mon,$year,
        $wday,$yday,$isday) = localtime (time);

print OUT " --> ",$year+1900,"/",$mon+1,"/",$mday+1;
print OUT " $hour:$min\n\n";
}

sub ifload {
my $sc_nr=shift;
my $x=shift;
my $y=shift;
my $hn=shift;
my $ip=shift;
my $if=shift;
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" ifDescr $if\nined -noupdate ";
$sc_string.="attribute \$stripchart$sc_nr ifLoad";
$sc_string.="{$if 0.00 %}\nined -noupdate ";
$sc_string.="label \$stripchart$sc_nr ifLoad\n";
stripchart ($sc_nr, $x, $y, $hn, $ip, $sc_string);
}

sub joblist {
# my @job_r=shift;
# my @job_s=shift;
# Jobmanagement for reachability
print OUT "set interpreter1 [ ined -noupdate create ";
print OUT "INTERPRETER ip_monitor.tcl ]\n";
foreach ( @jobr ) { print OUT $_; }

# Jobmanagement for SNMP monitoring
print OUT "\nset interpreter2 [ ined -noupdate create ";
print OUT "INTERPRETER snmp_monitor.tcl ]\n";
foreach ( @job ) { print OUT $_; }
}

#################################################################

# Verbindung zur Datenbank (sollte besser mittels Kommando-
# zeilenoption uebergeben werden)
use Mysql;
$DB = Mysql->connect("10.16.60.15", Hardwareverwaltung, hwvdb);


# X-Server
header ("server.tki");

$sth01=$DB->query("SELECT HostName,IPAddress FROM Rechner,Grouping
WHERE Grouping.GroupingID=21 AND Grouping.RechnerArtID=3 AND
Rechner.GroupingID=Grouping.GroupingID;");

$number=0;
```

```
$sc_nr=0;
$gn=0;
$nn=0;
$jid=0;
while ( @data = $sth01->fetchrow ) {
$rechner[$number]=$data[0];
$ipaddr[$number]=$data[1];
$number++; }

for ( $j=0; $j<$number; $j++) {
# definition of node (server)
$jobr[$jid]="ined send \$interpreter1 restart {Check ";
$jobr[$jid].="Reachability} 90.0 job$jid {\$node$nn}\n";
node ($nn++, 50, 60+$j*70, $rechner[$j], $ipaddr[$j],
"SUN-Server.xbm");
$jid++;

# number of logged in users on this server
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" HOST-RESOURCES-MIB!hrSystemNumUsers.0 ";
$sc_string.="{hrSystemNumUsers 0}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr ";
$sc_string.="HOST-RESOURCES-MIB!hrSystemNumUsers.0\n";
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_snmp_monitor 40 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" SNMPv2-MIB!sysUpTime.0";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.2.1.25.1.5.0\n";
stripchart ($sc_nr++, 118, 64+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);
$jid++;

# system load
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.="Monitor:RisingThreshold 1000\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.="UCD-SNMP-MIB!laLoadInt.1 {laLoadInt 200}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!laLoadInt.1\n";
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_snmp_monitor 40 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" SNMPv2-MIB!sysUpTime.0";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.10.1.5.1\n";
stripchart ($sc_nr++, 212, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);
$jid++;

# interface 1 load
$job[$jid]="ined send \$interpreter2 restart ";
```

```
$job[$jid].="start_ifload_monitor 30 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" \$stripchart$sc_nr 2";
ifload ($sc_nr++, 292, 64.00+$j*70, $rechner[$j], $ipaddr[$j],
"eth0");

# interface 2 load
$job[$jid].=" \$stripchart$sc_nr 3\n";
ifload ($sc_nr++, 372, 64.00+$j*70, $rechner[$j], $ipaddr[$j],
"eth1");
$jid++;

# diskusage for system data
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" Monitor:RisingThreshold 90\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" hrStorageDescr /\nined -noupdate attribute ";
$sc_string.="\$stripchart$sc_nr Monitor:ThresholdAction ";
$sc_string.="{flash write}\nined -noupdate attribute ";
$sc_string.="\$stripchart$sc_nr hrStorageLoad {/ 50.00 %}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.="hrStorageLoad\n";
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_stutil_monitor 120 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" \$stripchart$sc_nr 1";
stripchart ($sc_nr++, 452, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);

# diskusage for temp space
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" Monitor:RisingThreshold 80\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" hrStorageDescr /TMPDSK\nined -noupdate";
$sc_string.=" attribute \$stripchart$sc_nr Monitor:";
$sc_string.="ThresholdAction {flash write}\nined -noupdate";
$sc_string.=" attribute \$stripchart$sc_nr hrStorageLoad ";
$sc_string.="{/TMPDSK 50.00 %}\nined -noupdate label ";
$sc_string.="\$stripchart$sc_nr hrStorageLoad\n";
$job[$jid].=" \$stripchart$sc_nr 4\n";
stripchart ($sc_nr++, 532, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);
$jid++;

# group the last statements together
$ge="\$stripchart".($sc_nr-4)." \$stripchart".($sc_nr-3);
$ge.=" \$stripchart".($sc_nr-2)." \$stripchart".($sc_nr-1);
$label=$rechner[$j].": if/disk";
group ($gn++, 292.00, 64.00+$j*70, $label, $ge, "action.xbm");

# temperature 1
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
```

```
$sc_string.=" Monitor:RisingThreshold 50\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.=" -noupdate attribute \$stripchart$sc_nr";
$sc_string.="UCD-SNMP-MIB!extResult.1 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.1\n";
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_snmp_monitor 90 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" SNMPv2-MIB!sysUpTime.0";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.1";
stripchart ($sc_nr++, 372, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);
# temperature 2
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" Monitor:RisingThreshold 50\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.2 {extResult 20}\nined ";
$sc_string.="-noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.2\n";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.2";
stripchart ($sc_nr++, 452, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);

# cpu fan 1
$sc_string="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.4 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.4\n";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.4";
stripchart ($sc_nr++, 532, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);

# cpu fan 2
$sc_string="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.5 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.5\n";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.5\n";
stripchart ($sc_nr++, 612, 64.00+$j*70, $rechner[$j],
$ipaddr[$j], $sc_string);
$jid++;

# group the last statements together
$ge="\$stripchart".($sc_nr-4)." \$stripchart".($sc_nr-3);
```

```
$ge.=" \$stripchart".($sc_nr-2)." \$stripchart".($sc_nr-1);
$label=$rechner[$j].": temp/fan";
group ($gn++, 372.00, 64.00+$j*70, $label, $ge, "action.xbm");

}

joblist ( );

close (OUT);

# DXS

header ("dxs.tki");

# hier sollte besser die Position aus der DB bezogen werden, zu
# Demozwecken sind statische Eintraege vorgesehen worden

@groupid=(  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 16,
19, 20, 23, 30);
@ao=     (  3,  3,  4,  4,  4,  2,  4,  2,  8,  1,  6,  3,  5,
2,  3,  2,  3);
@gx_pos= (700,660,620,600,380,410,440,505,520,385,380,400,455,
710,380,520,540);
@gy_pos= (400,320, 95,150,250,350, 90,385,430,192,400,295,165,
220,140,255,320);
@x_pos=  (760,760,750,520, 30,230, 35,520,300, 90,100,  0,750,
630, 40,450, 28);
@y_pos=  (-10, 20, 20, 88,100,160,-25,300,400, 60,-30,  0,-20,
160, 70,170,-25);

$k=0;
$sc_nr=0;
$gn=0;
$tn=0;
$nn=0;
$jid=0;
foreach ( @groupid ) {

$sth01=$DB->query("SELECT HostName,IPAddress,Grouping.Name
FROM Rechner,Grouping WHERE Grouping.RechnerArtID=1 AND
Rechner.GroupingID=Grouping.GroupingID AND
Grouping.GroupingID=$_ ORDER BY Grouping.Name, HostName;");

$number=0;
while ( @data = $sth01->fetchrow ) {
$rechner[$number]=$data[0];
$ipaddr[$number]=$data[1];
$groupname=$data[2];
$number++; }


print OUT "set text$tn [ ined -noupdate create TEXT {Bereich:";
```

```
print OUT "$_} ]ined -noupdate move ";
print OUT "\$text$tn ",$x_pos[$k]," ",$y_pos[$k]-10;
print OUT "\nined -noupdate font \$text$tn fixed\n";
print OUT "ined -noupdate color \$text$tn Black\n\n";
$ge="\$text$tn ";
$tn++;

for ( $j=0; $j<$number; $j++) {
if ( $j%$ao[$k] == 0 ) { $y_pos[$k]+=60; }

# definition of node (dxs)
$jobr[$jid]="ined send \$interpreter1 restart {Check ";
$jobr[$jid].="Reachability} 90.0 job$jid {\$node$nn}\n";
$ge.="\$node$nn ";
node ($nn++, $x_pos[$k]+$j%$ao[$k]*110, $y_pos[$k],
$rechner[$j], $ipaddr[$j], "SparcStation.xbm");
$jib++;

# user activity
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" Monitor:RisingThreshold 50\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.1 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.1\n";
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_snmp_monitor 90 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" SNMPv2-MIB!sysUpTime.0";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.11";
$gr="\$stripchart$sc_nr ";
stripchart ($sc_nr++, $x_pos[$k]+$j%$ao[$k]*110, $y_pos[$k],
$rechner[$j], $ipaddr[$j], $sc_string);

# temperature 1
$sc_string="ined -noupdate attribute \$stripchart$sc_nr";
$sc_string.=" Monitor:RisingThreshold 50\n";
$sc_string.="ined -noupdate attribute \$stripchart$sc_nr ";
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.1 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.1\n";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.1";
$gr.="\$stripchart$sc_nr ";
stripchart ($sc_nr++, $x_pos[$k]+$j%$ao[$k]*110+220,
$y_pos[$k], $rechner[$j], $ipaddr[$j], $sc_string);

# cpu fan 1
$sc_string="ined -noupdate attribute \$stripchart$sc_nr ";
```

```
$sc_string.="Monitor:ThresholdAction {flash write}\nined";
$sc_string.="-noupdate attribute \$stripchart$sc_nr ";
$sc_string.="UCD-SNMP-MIB!extResult.4 {extResult 20}\n";
$sc_string.="ined -noupdate label \$stripchart$sc_nr";
$sc_string.=" UCD-SNMP-MIB!extResult.4\n";
$job[$jid].=" \$stripchart$sc_nr 1.3.6.1.4.1.2021.8.1.100.4\n";
$gr.="\$stripchart$sc_nr ";
stripchart ($sc_nr++, $x_pos[$k]+$j%$ao[$k]*110+330,
$y_pos[$k], $rechner[$j], $ipaddr[$j], $sc_string);
$jid++;

# interface 1 load
$job[$jid]="ined send \$interpreter2 restart ";
$job[$jid].="start_ifload_monitor 30 job$jid ";
$job[$jid].=$ipaddr[$j];
$job[$jid].=" \$stripchart$sc_nr 2";
$gr.="\$stripchart$sc_nr ";
ifload ($sc_nr++, $x_pos[$k]+$j%$ao[$k]*110+110, $y_pos[$k],
$rechner[$j], $ipaddr[$j], "eth0");
$jid++;

$gname=$rechner[$j].":data";
$ge.="\$group$gn ";
group ($gn++, $x_pos[$k]+54+$j%$ao[$k]*110, $y_pos[$k]+8,
$gname, $gr, "action.xbm");

}
group ($gn++, $gx_pos[$k], $gy_pos[$k], $groupname, $ge,
"bus.xbm");
$k++;
}

joblist ( );

close ( OUT );
```

F.3.2 Netzwerkmonitoring mittels "netsaint"

NetSaint[10] bietet eine webbasierte Überwachung vieler Netzwerkaspekte. Es integriert dabei ein Kollektor-Backend mit einem Webfrontend. Sowohl die Einstellungen für das Backend, als auch das Frontend lassen sich an die konkreten Anforderungen anpassen. Insgesamt müssen für NetSaint die Dateien *netsaint.cfg*, die zentrale Steuerung, *nscgi.cfg*, die Rechte und das Aussehen des Frontends, *commands.cfg*, die Kontroll- und Monitoringkommandos, *hosts.cfg*, die zentrale Rechner- und Dienstebeschreibung sowie *resource.cfg*, Beschreibung der Kontaktaufnahme, angepasst werden.
Das Tool **mk_netsaint** übernimmt dabei zwei Aufgaben. Es liest aus der

[10] siehe hierzu Abschnitt 17.6 und [W5]

Datenbank alle zu überwachenden Maschinen aus, bestimmt deren Typ und damit die Art der zu beobachtenden Dienste, welche die jeweilige Maschine anbietet. Die beobachteten Rechner werden wiederum zu Gruppen zusammengefaßt, die gemeinsam im Frontend angezeigt werden. Für jede Maschine und jede Kontaktperson oder Gruppe kann die Arbeitszeit eingetragen werden. So wird verhindert, dass Maschinen außerhalb ihrer Betriebszeiten fälschlicherweise offline gemeldet werden und abwesende Administratoren oder Diensteverantwortliche Fehlermeldungen erhalten. Dieses setzt jedoch die Hinterlegung entsprechender Informationen in der Datenbank voraus, wozu evtl. zusätzliche Felder eingefügt werden müssen.
Wie jede Maschine wird jede Kontaktperson einer Gruppe zugeordnet, welche die Ausfall- und Warnmeldungen der Rechner und Dienste erhält für die sie als verantwortlich eingetragen wurden. Hier können Abhängigkeiten definiert werden, so dass bei Unerreichbarkeit einer zentralen Netzwerkkomponente keine Tests mehr auf die "dahinterliegenden" Maschinen angesetzt werden. Zusätzlich zu diesen Verknüpfungen kann **mk_netsaint** in der *hosts.cfg* hinterlegen, wie schnell und wie häufig (bzw. sich steigernden Zeitspannen) die Warn- und Ausfallmeldungen übermittelt werden sollen. Eine solche Meldung kann beispielsweise per E-Mail oder mittels eines SMS-Gateways erfolgen. Die entsprechenden Konfigurationen findet man in der *resource.cfg*.
In der *commands.cfg* Datei werden die Überwachungskommandos definiert. Diese bestimmen welches Modul in welchen Zeitabständen zur Überprüfung einzelner Betriebszustände eingesetzt werden soll und welche Warn- bzw. Fehlermeldungen bei welchen Systemzuständen zurückgeliefert werden. Dabei können durchaus dieselben Plugins für unterschiedliche Rechner in verschiedenen Konfigurationen aufgerufen werden.
Die zweite von **mk_netsaint** generierte Datei: *nscgi.cfg* bestimmt Icons, welche für einezelne Hosts bzw. Dienste definiert werden können, sowie die Lage eines Rechners in der 2D- und 3D-Übersichtskarte. Dadurch erleichtert sich das Zurechtfinden im Webfrontend.

```
#!/usr/bin/perl -wT
#
# Skript zur Generierung der NetSaint-Konfiguration aus der
# Rechner/HardWare-DB, Version 0.2.8
#
# Dirk von Suchodoletz, <dirk@goe.net> 06-09-2001
#
################################################################

@sn=('Ping','SSHD running','KDM running','XFree running',
'AutoFS running','XDM running','500MB / free space',
'Logged on Users','NFS running','System Load H','System Load L');
@services=('POP3','IMAP','HTTP','FTP','DNS','SMTP','NFS','NIS');
```

```
# Verbindung zur Datenbank (fest kodiert, evtl. als Kommando
# zeilenoption vorsehen)
use Mysql;
$DB = Mysql->connect("134.76.60.35",Hardwareverwaltung,"hwvdb");

# Zieldateien oeffnen
open (HOSTS, ">hosts.cfg");
open (NSCGI, ">nscgi.cfg");

($S,$min,$hour,$mday,$mon,$year,
        $S,$S,$S) = localtime (time);

foreach $O ( 'HOSTS','NSCGI') {
print $O "# (c) Internet-AG 2001\n";
print $O "# (c) Dirk von Suchodoletz <dirk\@goe.net>,";
    print $O " (12-09-2001)\n#\n";
    print $O "# Automatically generated by mk_netsaint\n#";
    print $O " --> ",$year+1900,"/",$mon+1,"/",$mday+1;
    print $O " $hour:$min\n\n";
}

$sth01=$DB->query("SELECT GroupingID,Name,RechnerArtID FROM
Grouping ORDER by RechnerArtID;");

#################################################################
# Definitionen setzen
$num=0;
while ( @data = $sth01->fetchrow ) {
 $groupnr[$num]=$data[0];
 $groupnm[$num]=$data[1];
 $rechart[$num]=$data[2];
 SWITCH: {
  ($rechart[$num] eq "1") && do
{ $rechartnm[$num]="DXS"; $grnm[$num]="DXS_".$groupnr[$num];
    $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="hardware";
  $serv[$num][1][0]=$sn[0];
  $chkserv[$num][1][0]=";0;TP-".$groupnr[$num].";2;10;4;";
  $chkserv[$num][1][0].=$contact[$num].";180;TP-";
  $chkserv[$num][1][0].=$groupnr[$num].";1;1;0;;check_ping";
  $serv[$num][1][1]=$sn[1];
  $chkserv[$num][1][1]=";0;TP-".$groupnr[$num];
  $chkserv[$num][1][1].=";2;20;8;dxs-admin;300;TP-";
  $chkserv[$num][1][1].=$groupnr[$num].";1;1;0;;check-sshd";
  $serv[$num][1][2]=$sn[2];
  $chkserv[$num][1][2]=";0;TP-".$groupnr[$num];
  $chkserv[$num][1][2].=";2;15;8;dxs-admin;300;TP-";
  $chkserv[$num][1][2].=$groupnr[$num].";1;1;0;;check-kdm";
  $serv[$num][1][3]=$sn[3];
  $chkserv[$num][1][3]=";0;TP-".$groupnr[$num];
  $chkserv[$num][1][3].=";2;15;8;dxs-admin;300;TP-";
```

```
  $chkserv[$num][1][3].=$groupnr[$num].";1;1;0;;check-X";
  $serv[$num][1][4]=$sn[4];
  $chkserv[$num][1][4]=";0;TP-".$groupnr[$num];
  $chkserv[$num][1][4].=";2;15;8;dxs-admin;300;TP-";
  $chkserv[$num][1][4].=$groupnr[$num].";1;1;0;;check-automount";
 last SWITCH; };
 ($rechart[$num] eq "2") && do
{ $rechartnm[$num]="DXT"; $grnm[$num]="DXT_".$groupnr[$num];
  $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="hardware";
  $serv[$num][2][0]=$sn[0];
  $chkserv[$num][2][0]=";0;TP-".$groupnr[$num].";2;10;4;";
  $chkserv[$num][2][0].=$contact[$num].";180;TP-";
  $chkserv[$num][2][0].=$groupnr[$num].";1;1;0;;check_ping";
  $serv[$num][2][1]=$sn[5];
  $chkserv[$num][2][1]=";0;TP-".$groupnr[$num];
  $chkserv[$num][2][1].=";2;15;8;dxt-admin;300;TP-";
  $chkserv[$num][2][1].=$groupnr[$num].";1;1;0;;check-kdm";
  $serv[$num][2][2]=$sn[3];
  $chkserv[$num][2][2]=";0;TP-".$groupnr[$num];
  $chkserv[$num][2][2].=";2;15;8;dxt-admin;300;TP-";
  $chkserv[$num][2][2].=$groupnr[$num].";1;1;0;;check-X";
 last SWITCH; };
 ($rechart[$num] eq "3") && do
{ $rechartnm[$num]="X-Server"; $grnm[$num]="XSRV_".$groupnr[$num];
  $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="admin";
  $serv[$num][3][0]=$sn[0];
  $chkserv[$num][3][0]=";0;TP-".$groupnr[$num].";2;8;4;";
  $chkserv[$num][3][0].=$contact[$num].";120;TP-";
  $chkserv[$num][3][0].=$groupnr[$num].";1;1;1;;check_ping";
  $serv[$num][3][1]=$sn[1];
  $chkserv[$num][3][1]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][1].=";2;15;8;admin;300;TP-";
  $chkserv[$num][3][1].=$groupnr[$num].";1;1;1;;check-sshd";
  $serv[$num][3][2]=$sn[2];
  $chkserv[$num][3][2]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][2].=";2;15;8;admin;300;TP-";
  $chkserv[$num][3][2].=$groupnr[$num].";1;1;1;;check-kdm";
  $serv[$num][3][3]=$sn[8];
  $chkserv[$num][3][3]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][3].=";2;15;8;admin;300;TP-";
  $chkserv[$num][3][3].=$groupnr[$num].";1;1;1;;check_nfs";
  $serv[$num][3][4]=$sn[4];
  $chkserv[$num][3][4]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][4].=";2;15;8;admin;300;TP-";
  $chkserv[$num][3][4].=$groupnr[$num].";1;1;1;;check-automount";
  $serv[$num][3][5]=$sn[6];
  $chkserv[$num][3][5]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][5].=";2;20;8;admin;300;TP-";
```

```
  $chkserv[$num][3][5].=$groupnr[$num].";1;1;1;;df-500MB";
  $serv[$num][3][6]=$sn[7];
  $chkserv[$num][3][6]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][6].=";2;10;8;admin;300;TP-";
  $chkserv[$num][3][6].=$groupnr[$num].";1;1;0;;num-users";
  $serv[$num][3][7]=$sn[9];
  $chkserv[$num][3][7]=";0;TP-".$groupnr[$num];
  $chkserv[$num][3][7].=";2;10;8;admin;300;TP-";
  $chkserv[$num][3][7].=$groupnr[$num].";1;1;0;;sys-load-h";
 last SWITCH; };
 ($rechart[$num] eq "4") && do
{ $rechartnm[$num]="Workstation"; $grnm[$num]="LWS_".$groupnr[$num];
  $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="hardware";
  $serv[$num][4][0]=$sn[0];
  $chkserv[$num][4][0]=";0;TP-".$groupnr[$num].";2;15;8;";
  $chkserv[$num][4][0].=$contact[$num].";240;TP-";
  $chkserv[$num][4][0].=$groupnr[$num].";1;1;0;;check_ping";
  $serv[$num][4][1]=$sn[1];
  $chkserv[$num][4][1]=";0;TP-".$groupnr[$num];
  $chkserv[$num][4][1].=";2;20;8;admin;300;TP-";
  $chkserv[$num][4][1].=$groupnr[$num].";0;0;0;;check-sshd";
  $serv[$num][4][2]=$sn[4];
  $chkserv[$num][4][2]=";0;TP-".$groupnr[$num];
  $chkserv[$num][4][2].=";2;15;8;admin;300;TP-";
  $chkserv[$num][4][2].=$groupnr[$num].";0;0;0;;check-automount";
  $serv[$num][4][3]=$sn[3];
  $chkserv[$num][4][3]=";0;TP-".$groupnr[$num];
  $chkserv[$num][4][3].=";2;20;8;admin;300;TP-";
  $chkserv[$num][4][3].=$groupnr[$num].";0;0;0;;check-X";
  $serv[$num][4][4]=$sn[2];
  $chkserv[$num][4][4]=";0;TP-".$groupnr[$num];
  $chkserv[$num][4][4].=";2;20;8;admin;300;TP-";
  $chkserv[$num][4][4].=$groupnr[$num].";0;0;0;;check-kdm";
 last SWITCH; };
 ($rechart[$num] eq "5") && do
{ $rechartnm[$num]="Server"; $grnm[$num]="SRV_".$groupnr[$num];
  $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="admin";
  $serv[$num][5][0]=$sn[0];
  $chkserv[$num][5][0]=";0;TP-".$groupnr[$num].";2;10;4;";
  $chkserv[$num][5][0].=$contact[$num].";60;TP-";
  $chkserv[$num][5][0].=$groupnr[$num].";1;1;1;;check_ping";
  $serv[$num][5][1]=$sn[1];
  $chkserv[$num][5][1]=";0;TP-".$groupnr[$num];
  $chkserv[$num][5][1].=";2;15;8;admin;300;TP-";
  $chkserv[$num][5][1].=$groupnr[$num].";1;1;0;;check-sshd";
  $serv[$num][5][2]=$sn[10];
  $chkserv[$num][5][2]=";0;TP-".$groupnr[$num];
  $chkserv[$num][5][2].=";2;10;8;admin;300;TP-";
```

```
  $chkserv[$num][5][2].=$groupnr[$num].";1;1;0;;sys-loadl";
 last SWITCH; };
 ($rechart[$num] eq "6") && do
{ $rechartnm[$num]="Switch"; $grnm[$num]="MSW_".$groupnr[$num];
  $cmdline[$num]="check-host-alive;3;240;";
  $cmdline[$num].="TP-".$groupnr[$num].";1;1;1;";
  $contact[$num]="netzwerk";
  $serv[$num][6][0]=$sn[0];
  $chkserv[$num][6][0]=";0;TP-".$groupnr[$num].";2;10;4;";
  $chkserv[$num][6][0].=$contact[$num].";60;TP-";
  $chkserv[$num][6][0].=$groupnr[$num].";1;1;0;;check_ping";
 last SWITCH; };
}
$num++;
}

#####################################################################
# prepare contact group entries
$cgre="";
$sth02=$DB->query("SELECT Name,Name_Lang,MKontaktID FROM
MKontakt WHERE KontaktArtID>2;");
$i=0;
while ( @data = $sth02->fetchrow ) {
$cgrl[$i][0]=$mkid=$data[2];
$_=$data[0];
$0=tr/A-Z/a-z/;
$cgrl[$i][1]=$name=$_;
$cgre.="contactgroup[".$name."]=".$data[1].";";
$con[$i]="contact[".$name."]=".$data[1].";TP-";
$con[$i].=$name.";TP-".$name.";";
$sth03=$DB->query("SELECT Value FROM MKontakt_Properties,
MKontakt WHERE PropertyNameID=88 AND
MKontakt_Properties.MKontaktID=$mkid
AND MKontakt_Properties.MKontaktID=MKontakt.MKontaktID;");
$j=0;
while ( @data = $sth03->fetchrow ) {
$kid=$data[0];
$sth04=$DB->query("SELECT Name,Name_Lang FROM MKontakt
WHERE MKontaktID=$kid;");
if ( @data = $sth04->fetchrow ) {
$_=$data[0];
$0=tr/A-Z/a-z/;
$cgre.=$_.",";
$j++; }
}
if ($j==0) {$cgre.=$name;}
else { chop ($cgre); }
$cgre.="\n";
$i++;
}

#####################################################################
```

```
# process online hours for timeperiod statement
$i=0;
print HOSTS "#\n# timeperiod definitions\n#\n";
while ($i<$num) {
$gid=$groupnr[$i];
print HOSTS "timeperiod[TP-".$groupnr[$i]."]=Opening Hours of ";
print HOSTS $groupnm[$i].";";
$j=73; $tp='';
while ($j<80) {
$sth02=$DB->query("SELECT Value FROM Grouping_Properties
WHERE PropertyNameID=$j AND GroupingID=$gid;");
if ( @data = $sth02->fetchrow ) {
$tp.=$data[0].";"; }
else { $tp.=";"; }
$j++;
}
chop ($tp);
print HOSTS $tp."\n";
$i++; }
$i=0;
while ($cgrl[$i][0]) {
print HOSTS "timeperiod[TP-".$cgrl[$i][1];
print HOSTS "]=Work Hours of ".$cgrl[$i][1].";";
$j=73; $tp='';
while ($j<80) {
$kid=$cgrl[$i][0];
$sth02=$DB->query("SELECT Value FROM MKontakt_Properties
WHERE PropertyNameID=$j AND MKontaktID=$kid;");
if ( @data = $sth02->fetchrow ) {
$tp.=$data[0].";"; }
else { $tp.=";"; }
$j++;
}
chop ($tp);
print HOSTS $tp."\n";
$i++;
}

##################################################################
# process hosts in the several groups
$i=0;
print HOSTS "#\n# host definitions\n#\n";
print NSCGI "#\n# EXTENDED HOST INFORMATION\n#\n";
while ($i<$num) {
$gid=$groupnr[$i];
$sth02=$DB->query("SELECT HostName,RechnerID,IPAddress FROM
Rechner WHERE GroupingID=$gid ORDER by IPAddress;");
print HOSTS "# hosts -> ".$groupnm[$i]."\n";
$j=0;
while ( @data = $sth02->fetchrow ) {
$rechlist[$i][$j]=$data[0];
$rid=$data[1];
```

```
print HOSTS "host[".$data[0]."]=".$rechartnm[$i]."/SN #";
print HOSTS $rid.";".$data[2].";;".$cmdline[$i]."\n";
print NSCGI "hostextinfo[".$data[0]."]=;".$rechartnm[$i];
print NSCGI ".gif;".$rechartnm[$i].".gif;".$rechartnm[$i];
print NSCGI ".gd2;".$data[0]."/SN #".$rid.";";
$k=80; $tp='';
while ($k<85) {
$sth03=$DB->query("SELECT Value,PropertyNamen.Name
FROM Rechner_Properties,PropertyNamen WHERE
RechnerID=$rid AND PropertyNamen.PropertyNameID=$k
AND PropertyNamen.PropertyNameID=
Rechner_Properties.PropertyNameID;");
if ( @data = $sth03->fetchrow ) {
$tp.=$data[0]; }
else { $tp.="0"; }
if ($k==81 || $k==84) { $tp.=";"; }
else { $tp.=","; }
$k++;
}
chop ($tp);
print NSCGI $tp."\n";
$j++;
}
$i++;
}
#####################################################################
# process host groups
$i=0;
print HOSTS "#\n# hostgroup definitions\n#\n";
while ($i<$num) {
$tp="hostgroup[".$grnm[$i]."]=".$groupnm[$i].";";
if ( $rechlist[$i][0] ) {
print HOSTS $tp.$contact[$i].";".$rechlist[$i][0];
$j=1;
while ( $val=$rechlist[$i][$j] ) {
print HOSTS ",$val"; $j++;
}
print HOSTS "\n"; }
$i++;
}
# process standard host services
$i=0;
print HOSTS "#\n# service definitions\n#\n";
while ($i<$num) {
print HOSTS "#\n# hostgroup: ".$groupnm[$i]."\n#\n";
$j=0;
while ($rechlist[$i][$j]) {
$k=0;
print HOSTS "# host -> ".$rechlist[$i][$j]." (Typ: ";
print HOSTS $rechartnm[$i].")\n";
while ($chkserv[$i][$rechart[$i]][$k]) {
print HOSTS "service[".$rechlist[$i][$j]."]=";
```

```
print HOSTS $serv[$i][$rechart[$i]][$k];
print HOSTS $chkserv[$i][$rechart[$i]][$k]."\n";
$k++; }
foreach $S (@services) {
$hn=$rechlist[$i][$j];
$_=$S;
$O=tr/A-Z/a-z/;
$sth01=$DB->query("SELECT HostName FROM Rechner_Properties,
Rechner,PropertyNamen WHERE Name='$S' AND HostName='$hn'
AND PropertyNamen.PropertyNameID=
Rechner_Properties.PropertyNameID
AND Rechner_Properties.RechnerID=Rechner.RechnerID
ORDER by HostName;");
while (@data = $sth01->fetchrow) {
print HOSTS "service[".$hn."]=Check $S;0;TP-";
print HOSTS $groupnr[$i].";2;20;8;admin;240;TP-";
print HOSTS $groupnr[$i].";1;1;0;;check_$_\n";
$addep[$i][$j][$k]="servicedependency[".$rechlist[$i][$j];
$addep[$i][$j][$k].=";Check $S]=".$rechlist[$i][$j].";";
$addep[$i][$j][$k].=$serv[$i][$rechart[$i]][0].";c;wc\n";
$k++;}
}
$j++; }
$i++; }
# process non standard host services
print HOSTS "#\n# non standard service definitions\n#\n";

#################################################################
# print out contacts
$i=0;
print HOSTS "#\n# contact definitions\n#\n";
while ($con[$i]) {
print HOSTS $con[$i]."1;1;1;1;1;1;notify-by-email;"
print HOSTS "host-notify-by-email;dirk\@goe.net;123456\n";
$i++;
}


# print out contact groups
print HOSTS "#\n# contact group definitions\n#\n";
print HOSTS $cgre;

# process host services dependencies to avoid overhead
$i=0;
print HOSTS "#\n# service dependecy definitions\n#\n";
while ($i<$num) {
print HOSTS "#\n# hostgroup: ".$groupnm[$i]."\n#\n";
$j=0;
while ($rechlist[$i][$j]) {
$k=1;
print HOSTS "# host -> ".$rechlist[$i][$j]." (Typ: ";
print HOSTS $rechartnm[$i].")\n";
```

```
while ($serv[$i][$rechart[$i]][$k]) {
print HOSTS "servicedependency[".$rechlist[$i][$j];
print HOSTS ";".$serv[$i][$rechart[$i]][$k]."]=";
print HOSTS $rechlist[$i][$j].";";
print HOSTS $serv[$i][$rechart[$i]][0].";c;wc\n";
$k++;}
while ($addep[$i][$j][$k])
{ print HOSTS $addep[$i][$j][$k]; $k++; }
$j++;
}
$i++;
}

# close files
close (HOSTS);
close (NSCGI);
```

F.3.3 Weitere Überwachungstools

Als einfache Tests ohne großen Overhead an Funktionalität kommen Erreichbarkeitsüberprüfungen mittels "Ping" infrage. Diese ließen sich als Shell- oder Perl-CGI-Skripten schnell und problemlos erstellen und mit der Datenbank verknüpfen. Diese Programme könnten, etwas Geduld seitens der Administratoren vorausgesetzt, ihre Ergebnisse "on-the-fly" generieren. Eine etwas tiefgehendere Analyse könnte durch das Auslesen der Uptime-Variablen der SNMP-Schnittstelle erfolgen, wobei hier aufgrund längerer Laufzeiten bis zur Antwort eine Ad-Hoc-Generierung der Zustandsdaten nicht mehr sinnvoll erscheint.

Anhang G

Fehlersuche

G.1 Generelle Gedanken

Auf festplattenlosen Systemen gestaltet sich die Fehlersuche etwas schwieriger; es muss bereits fast alles, insbesondere jedoch die Netzwerkkonfiguration funktionieren, bevor ein Logservice gestartet werden kann. Eine gute Quelle für auftretende Fehler liegt im Serverlogfile. Wenn man auf dem Thin-Client den **sshd** startet, kann ein Teil der Fehlersuche, gerade bei der Konfiguration des X11 auch remote erfolgen.
Die vorstellbaren Fehler werden in den nächsten Abschnitten in der Reihenfolge ihres Auftretens abgehandelt und so weit es geht, Anleitungen zu ihrer Behebung angegeben.
Die grundsätzliche Konfiguration eines jeden Client-Types (Diskless X-Station oder X-Terminal) soll so vollständig wie möglich über die verschiedenen Einträge in der Konfiguration des DHCP erfolgen, um einen möglichst einfachen Überblick behalten zu können. Bestimmte host- oder devicespezifische Dateien sollten soweit es geht vermieden werden, weil diese bei Anpassungen oder Updates häufig vergessen werden und damit eine häufige Fehlerursache bilden.

G.2 Boot-ROM oder -Code wird nicht erkannt

Wird nach dem Starten des Rechners und dem regulären Inititalisieren des BIOS bis zum Booten von Festplatte, CD-ROM oder Diskette der Bootcode nicht ausgeführt, kann es daran liegen, dass die EPROM-Option der Netzwerkkarte nicht konfiguriert wurde. Einige BIOS'es bieten die Möglichkeit eine Netzboot-Option explizit einzuschalten.

Läuft alles korrekt, meldet sich Etherboot mit seiner Versionsnummer, gibt den gefundenen Typ von Netzwerkkarte an und liefert die MAC-Adresse der Karte. Bei Netboot meldet sich der Netbootloader und anschließend der Packetdriver der Netzwerkkarte. Wird PXE vorgeschaltet, müssen zuerst dessen Meldungen ausgewertet werden, bevor die Funktion von Etherboot überprüft wird.

ISA-Netzwerkkarten Viele ältere ISA-Netzwerkkarten unterstützen nur kleine EPROMs, d.h. 8 kByte (27C64)[1], 16 kByte (27C128) und manche auch noch 32 kByte (27C256). Hier sollten die Herstellerangaben überprüft werden. Es gilt jedoch die Faustregel: Je älter oder preiswerter eine Netzwerkkarte war, desto kleiner sind die unterstützten EPROM-Größen. Ältere Novell-NE2000-Karten können teilweise nur mit maximal 16k Byte großen ROMs umgehen.
Häufig sind für ältere Mainboards die Ressourcen manuell zu verteilen, d.h. der Systemadministrator muss selbst darauf achten, dass keine Interuptkanäle, DMA-Leitungen oder IO-Bereiche der verwendeten Erweiterungskarten sich überschneiden.
Weitere Schwierigkeiten verbergen sich hinter Jumper-Einstellungen, Plug'n Play oder der Software-Konfiguration einzelner Karten. Plug'n Play sollte man versuchen zu vermeiden, da man dann nur noch sehr wenig Einfluss auf die tatsächlichen Einstellungen der Karte hat, so dass die EPROM-Option häufig ausgeschaltet bleibt[2].
Sind die Netzwerkkarten jumperless wird sind die notwendigen Diagnosewerkzeuge zu beschaffen, um die Konfiguration zu überprüfen. Einige Netzwerkkarten lassen sich auch unter Linux konfigurieren[3]. Vielfach wird man mit mehreren Einstellungen experimentieren müssen, da nicht jedes ROM in jeder denkbaren Einstellung erkannt wird. Wenn ein 16 kByte Etherbootimage in einem 32 kByte ROM liegt, funktioniert manchmal nur das Setzen der ROM-Größe auf 16 kByte.
Der Bootcode wird im Gegensatz zur PCI-Version immer, wenn er vom BIOS als ROM-Extension erkannt wird, ausgeführt. Ist die Netzwerkkarte nicht unterstützt wird die Meldung "probing [XYZ]" und dann "no adaptor found" ausgegeben.

PCI-Netzwerkadapter Sind PCI-Netzwerkkarten im System installiert, kann es dazu kommen, dass sich Etherboot nicht meldet: Dann stimmen die

[1] Typenbezeichnungen, siehe Abschnitt zu (E)EPROMs

[2] Diese Erfahrungen werden auf Etherboot-Mailingliste häufiger berichtet. Siehe hierzu [W3].

[3] Donald Becker hat für einige Netzwerkkarten **netdiag** geschrieben.

PCI- und Vendor-ID's der Netzwerkkarte nicht mit dem erstellten Etherbootcode überein. Hierzu sollte man das README von Etherboot bemühen und überprüfen, welcher Treiber für welchen Chipset verwendet werden sollte. Nicht alle DLink-Netzwerkkarten verwenden z.B. Tulip-Chipsets.
In den meisten Fällen muss die Option "Boot-ROM" mit dem Diagnose- oder Setuptool der Netzwerkkarte eingeschaltet werden, selbst wenn die Software dem Mainboard-BIOS hinzugefügt wurde. Einige ältere BIOS-Versionen unterstützen das in einem früheren Kapitel beschriebene Patchen nicht, hier kann man versuchen sich mit einem Update zu behelfen oder muss auf externe ROMs ausweichen.

G.3 Bootcode ausgeführt - "<sleep>"

Wird der Bootcode erfolgreich abgearbeitet, auf dem Bildschirm scrollt hingegen nur die Ausgabe "<sleep>" über den Monitor, gibt es hierfür mehrere Ursachen:
Evtl. wurde nicht der richtige Medienanschluss auf der älteren ISA-Netzwerkkarte über das Setup-Utility oder entsprechende Jumper ausgewählt. Ältere 3COM-Combo-Karten realisieren z.B. kein Autosense des angeschlossenen Medientyps. Auf der Serverseite kann z.B. in der */etc/dhcpd.conf* nicht die korrekte MAC eingetragen worden sein. Oder nach einer Änderung der Optionen wurde der **dhcpd** oder **bootpd** nicht neu gestartet. In beiden Fällen kann **tcpdump** helfen, nachzusehen ob überhaupt Pakete vom Client versandt werden. Ein sehr gutes Werkzeug mit grafischer Oberfläche liegt mit **ethereal** vor, welches tiefe Einblicke in die einzelnen Protokollschichten erlaubt. Hierzu ist es natürlich am Besten in einem Netzwerk zu arbeiten, in dem so wenig wie möglich zusätzlicher Traffic herrscht. Im besten Fall baut man sich einfach eine Testumgebung mit zwei Rechnern, die über einen Hub verbunden sind, auf.
Mit dem Kommando **bootpc** kann die Funktion von BOOTP- und DCHP-Servern überprüft werden: Mit der Option "–hwaddr" kann man die Einstellungen für einen bestimmten Client testen, "–verbose" erhöht den Umfang der Ausgabe. **bootpc** ist Bestandteil des BOOTP-Paketes. Das Programm **dhcptest** bietet sich auch zum Testen des BOOTP/DHCP-Dienstes an. Hiermit lassen sich zusätzlich alle DHCP-typischen Datenfelder und Dienste abfragen.

G.4 IP-Konfiguration erfolgte, TFTP reagiert nicht

Werden zwar die IP-Parameter richtig übermittelt, aber das Kernelimage nicht übertragen, muss untersucht werden, ob der TFTP-Daemon auf dem Server läuft und über die entsprechenden Pfade und Rechte verfügt. Evtl. sollte man das Logging dieses Daemons wärend des Debugging hochsetzen, um im */var/log/messages* Fehlermeldungen über falsch angeforderte Dateien, der Filename des Netkernels wurde per BOOTP/DHCP falsch übermittelt, oder verletzte Rechte zu erfahren. Leider kann man auf Clientseite häufig nicht viel erkennen, da die Fehlerausgaben sehr schnell erfolgen und eine erneute Übertragung des Kernels immer wieder angefordert wird. An dieser Stelle sollte dann eine Untersuchung mit **ethereal** erfolgen. Probleme können im Zusammenhang mit PXE auftreten, wenn der TFTP-Daemon bestimmte Optionen nicht unterstützt. Diesem Problem kann unter Umständen aus dem Weg gegangen werden, wenn Etherboot NFS[4] zum Kopieren des Kernels über das Netzwerk verwendet.

G.5 Kernel übertragen

Wenn der Kernel auf der Clientmaschine erfolgreich gestartet werden konnte, bekommt man meistens schon einen Hinweis, wo der nächste Fehler liegen könnte. Der Kernel merkt eine unvollständige IP-Konfiguration an, wenn diese nicht korrekt von der Bootsoftware bezogen werden konnte. Hier liegt eventuell das Problem vor, dass **mknbi(-linux)** nicht mit der Option "–ipaddrs=rom" aufgerufen wurde und so die durch Etherboot bezogene IP-Konfiguration nicht an den Kernel übergeben wird. Häufig wird vergessen, die verwendete Netzwerkkarte in den Kernel zu kompilieren, oder diese wird aus irgendeinem Grund vom Kernel nicht erkannt. Der Netzwerkkartentreiber von Etherboot spielt zu diesem Zeitpunkt keine Rolle mehr und wird beim Dekomprimieren des Kernels abgeschaltet.

Netzwerkkartenprobleme Letzeres passiert bei ISA-Karten, wenn ihre Konfiguration außerhalb des Autoprobingbereichs[5] des Kernels liegt. Bei PCI-Karten kann es zu IRQ-Konflikten durch IRQ-Sharing auf dem Bus kommen. Dann kann man versuchen, dieses durch Umstecken der Karte zu umgehen. An dieser Stelle noch einmal der Hinweis: Das erfolgreiche

[4] Siehe hierzu die Ausführungen zu den Compile-Time-Options von Etherboot (Abschnitt B.2)

[5] meistens werden IO-Adressen im Bereich von 0x260 bis 0x360 gescannt

Booten des Kernels sagt nichts darüber aus, dass anschließend weiterhin die Netzwerkkarte funktioniert, da dann der Kernel vom Bootcode die Kontrolle übernommen hat.

Konfigurationsfehler Ein weiterer Fehler kann beim Mounten des Rootfilesystems auftreten. Entweder es klappt bereits nicht wegen o.g. Problems der IP-Konfiguration oder der Netzwerkkarte oder es scheitert am Parameter für das Rootfilesystem. Durch **mknbi(-linux)** oder die entsprechende DHCP-Option wird dem Kernel der Pfad mitgeteilt, von wo er das Rootfilsystem mounten soll. Die IP-Nummer des Servers bezieht der Client von der Bootsoftware, der Pfad wird beim "taggen" des Kernels[6] angegeben. Eine falsche Pfadangabe kann aber schon anhand der Fehlerausgaben des Kernels identifiziert werden.

Kernel wurde unvollständig erstellt Kernelfehler können sich an den verschiedensten Stellen auswirken: Das Fehlen der entsprechenden Netzwerkkarte oder fehlende Parameter bei der Übergabe wurden schon angesprochen. Weitere Fehler treten auf, wenn die benötigten Filesysteme NFS, Ramfile, Minix mit Ramdisk nicht fest in den Kernel eingebunden wurden. Dieses macht sich bemerkbar, wenn bereits das Rootfilesystem nicht gemountet wird oder "permission denied"-Meldungen erscheinen, wenn die Ramdisk mit ihrer Verzeichnisstruktur und Daten gefüllt werden soll. In letzterem Fall fährt das System zwar in gewisser Weise hoch, ein Einloggen, bzw. Arbeiten damit, ist jedoch nicht möglich, da nicht nur die Rechte der Devices nicht stimmen.
Es können also nur Treiber in Module ausgelagert werden, die nicht zwingend zum Start des Thin-Clients notwendig sind. Fehlende Module verhindern später vielleicht das Funktionieren des Grafiktreibers (XFree86), da die AGP-Gart-Unterstützung oder die serielle Maus nicht geladen werden konnten. Ein Einloggen auf dem System und manuelle Fehlersuche und -beseitigung sind jedoch in diesem Stadium bereits realisierbar.

G.6 NFS-Probleme

Auch beim Mounten des NFS-Filesystems kann einges fehlschlagen: Entweder ist die Freigabe in der */etc/exports* nicht korrekt oder zu restriktiv oder der Client nicht dem DNS bekannt, so dass aus diesen Gründen ein Zugriff verweigert wird. Ein Blick ins */var/log/messages* schafft an diesem Punkt meistens den notwendigen Durchblick. Am einfachsten überprüft man die

[6] Siehe hierzu den Abschnitt B.4

Funktionsfähigkeit von NFS, in dem man von einer normalen Workstation aus versucht, das Rootfilesystem und bei DXS die weiteren Freigaben zu mounten. Bei den meisten NFS-Implementierungen ist ein funktionierender Domain Name Service (DNS) zwingende Voraussetzung, d.h. die mountenden Systeme sollten dem NFS-Server bekannt sein.
Aus Performancegründen wird man versucht sein unter Linux den Kernelnfsd zu verwenden. Dieser ist in der Kernel-Version 2.2.X jedoch nicht vollständig stabil und bringt bei ReadWrite-gemounteten Shares häufiger Probleme mit sich: Ein Einbinden dieser Shares funktioniert häufig erst wieder nach einem Neustart des Daemons. Für die ReadOnly-Bereiche kann der Einsatz jedoch vorteilhaft sein. Mit den Kernelversionen ab 2.4.X gibt es einige Verbesserungen, jedoch muss man auch hier auf einige Ungereimtheiten gefaßt sein, da die Codebasis noch nicht vollständig gefestigt ist.

G.7 Init-Fehler

Alles bisherige lief erfolgreich, aber es folgt nun die Ausgabe "unable to open initial console" oder "no init found". Diese Fehler liegen im unvollständigen Filesystem oder NFS-Problemen begründet. Wird das NFS von einer Maschine mit anderer Architektur[7] gemountet, kann es Fehler bei den Major- und Minornumbers der Deviceeinträge geben. Dieses ermittelt man am leichtesten, indem man auf einer Linuxworkstation des gleichen Architekturtypes wie der Thin-Client das Rootfilesystem mountet.
Weiterhin kann es passieren, dass die Dateirechte nicht stimmen: Das Rootfilesystem wird mit "(ro)" sehr restriktiv exportiert. Alle Rootrechte werden auf "nobody" gemappt, d.h. bei allen Dateien, die Root ausführen soll, muss das Executebit für "other" gesetzt sein. Unter Abwägung verschiedener Sicherheitsaspekte[8] sollte die Freigabe evtl. durch die Option "no_root_squash" erweitert werden. Für die Zeit der Einrichtung der Thin-Clients und wärend des Debuggings kann das NFS-Export auf "(rw, no _root_squash)" gesetzt werden. Dadurch entstehen aber größere Sicherheitslöcher und die Clients können bei fehlerhafter Konfiguration sich gegenseitig ihre Einstellungen überschreiben.

G.8 Skriptfehler

Neben wirklichen Programmierfehlern in Skripten tritt häufig ein Fehlverhalten auf, wenn bestimmte Filesysteme oder Treiberunterstützungen im

[7] 64 bit statt 32 bit, "little endian" statt "big endian"
[8] Siehe hierzu Abschnitt 8.4

Kernel nicht vorliegen. Filesystemfehler wurden bereits erörtert, für die Konfiguration via **dhclient** ist die Unterstützung von Sockets und ihrer Filterung im Kernel notwendig. Ob **dhclient** erfolgreich ausgeführt werden konnte, kann man anhand des Inhaltes von */var/state/dhcp/dhclient.leases* überprüfen. Hier sollten alle Variablen deklariert sein, wie sie in */etc/dhclient.conf* angefordert und auf dem Server in */etc/dhcpd.conf* definiert wurden. Stehen mehrere DCHP-Server zur Auswahl, kann anhand der Variablen "dhcp-server-identifier" bestimmt werden, woher der Client seine Konfiguration bezogen hat. Es ist bei mehreren Servern durchaus wahrscheinlich, dass der ursprüngliche Bootserver, von dem der Kernel und das Rootfilesystem bezogen werden und der Server mit den Konfigurationsdaten voneinander abweichen können, wenn nicht im **boot**-Skript ein dedizierter Server angegeben wird.
Läuft der Bootvorgang bis zum Aufruf von **dhclient** normal und bricht danach die Netzwerkverbindung ab, liegt ein schwerwiegenderes Problem in **dhclient-script** vor. Dann sollte der entsprechende Aufruf in der */etc/init.d/boot* auskommentiert werden und **dhclient** manuell gestartet werden, um den Fehler eingrenzen zu können.
dhclient-script übernimmt vielfältige Konfigurationsaufgaben, weshalb bei Fehlern in der *(/nfsroot/dxs)/etc/(X11/)/XF86Config*, *(/nfsroot/dxs)/etc/rc.config*, *(/nfsroot/dxs)/etc/resolv.conf* ... dieses Skript zu untersuchen ist. An dieser Stelle muss weiterhin überprüft werden, ob alle notwendigen Variablen vollständig und bei Strings in der richtigen Reihenfolge innerhalb des Strings übermittelt werden. So unterscheiden sich z.B. die entsprechenden Anpassungen zum Erzeugen der *XF86Config* für XFree86 Version 3.3.6 und 4.X.Y.
Die *(/nfsroot/dxs)/etc/rc.config* spielt eine zentrale Rolle beim Starten der verschiedenen Dienste und sollte beim Auftreten von Fehlern in diesem Bereich untersucht werden. Möchte man nicht alle Variablen für die verschiedenen Dienste in der */etc/dhcpd.conf* auf dem Server definieren, kann man sinnvolle Standardwerte in der *(/nfsroot/dxs)/etc/rc.config.default* oder *(/nfsroot/dxs)/etc/dhclient.conf* eintragen.

G.9 Redundante Server

Zur Erreichung von Redundanz bzw. als Fail-Over-Lösung im Serverbetrieb wird man in vielen Fällen mehrere Server betreiben, welche DHCP, TFTP und NFS anbieten. Hier können jedoch Probleme auftreten, wenn diese Server nicht einheitlich konfiguriert sind. Das Verhalten der einzelnen Clients wird dann vom jeweiligen Server abhängig, was eine Fehlersuche durch scheinbare Verwirrung des Setups erschweren kann. Daher sollte nicht

nur überprüft werden, von welcher Maschine der Thin-Clien ursprünglich bootete und den Kernel bezog, sondern woher die weiteren Konfigurationsparameter stammen. Dieses kann der Datei */var/state/dhcp/dhclient.leases* entnommen werden.

Anhang H

Quellen

Monografien:

[M1] Comer, Douglas E., *TCP/IP: Principles, Protocols, and Architectures*, 4th edition, Prentice Hall, Upper Saddle River 2000.

[M2] Dawson, Terry and Kirch, Olaf, *Linux - Wegweiser für Netzwerker*, 2. Auflage, O'Reilly, Beijing, Cambridge u.a. 2001.

[M3] Ralph Droms,Ted Lemon, *The DHCP Handbook: Understanding, Deploying, and Managing Automated Configuration Services*, New Riders Publishing, 1999.

[M4] Dubois, Paul, *MySQL - Entwicklung, Implementierung und Referenz*", Markt+Technik Verlag, München 2000.

[M5] Hantelmann, Fred, *Linux für Durchstarter*, Springer, Berlin 1997.

[M6] Hetze, Hohndel u.a. *Linux: Anwender Handbuch*, 7. Auflage, LunetIX, Berlin 1997.

[M7] Kofler, Michael, *Linux: Installation, Konfiguration, Anwendung*, 3. Auflage, Addison-Wesley-Longman, Bonn u.a. 1998.

[M8] Krienke, Rainer, *Programmieren in Perl*, Hanser, München 1998.

[M9] Yarger, Reese, King, *MySQL & mSQL*, O'Reilly, Beijing, Cambridge u.a. 1999.

Zeitschriften:

[Z1] Bauer, Günter, "Über 100 Uni-Rechner im Griff", *NetworkWorld* 26. Oktober 2001, Seite 28.

[Z2] Fischbach, Rainer, "New Economy: Wie produktiv die Informationstechnik wirklich ist", *iX* 09/2001, Seiten 100,101.

[Z3] Hantelmann et. al., "Jenseits des PC's", *iX* 03/2000, Seiten 56-68,130-136.

[Z4] Ritter, Marcel, "Automatische Installation und Konfiguration von Linux", *Linux-Magazin* 09/2001, Seite 97-101.

[Z5] von Suchodoletz, D., "Gut gebootet, Linux X-Terminals", *Linux-Magazin* 08/1999, Seite 101-110.

[Z6] von Suchodoletz, D., "Thin-Clients - Plattenloser Arbeitsplatz selbstgemacht", *Linux-Magazin* 08/2000, Seite 110-115.

[Z7] Kulisch, Meyer, Steffens, "Ausgetauscht, Linux auf 3000 verteilten Arbeitsplätzen", *iX* 03/2002, Seite 40 ff.

Web-Links:

[W1] http://www.thinguin.org
[W2] http://www.rom-o-matic.net
[W3] http://etherboot.sourceforge.net
[W4] http://netboot.sourceforge.net
[W5] http://netsaint.sourceforge.net
[W6] http://www.isc.org
[W7] http://wwwhome.cs.utwente.nl/ schoenw/scotty
[W8] http://www.mysql.org
[W9] http://www.postgresql.org
[W10] http://www.ltsp.org
[W11] http://www.escape.de/users/outback/linux/dlc.html

[W12] http://ldc.goe.net

Index

In der Reihe GWDG-Berichte sind zuletzt erschienen:

Nähere Informationen finden Sie im Internet unter
`http://www.gwdg.de/forschung/publikationen/gwdg-berichte`

Nr. 40 *Plesser, Theo und Peter Wittenburg* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1994
1995

Nr. 41 *Brinkmeier, Fritz* (Hrsg.):
Rechner, Netze, Spezialisten. Vom Maschinenzentrum zum Kompetenzzentrum - Vorträge des Kolloquiums zum 25jährigen Bestehen der GWDG
1996

Nr. 42 *Plesser, Theo und Peter Wittenburg* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1995
1996

Nr. 43 *Wall, Dieter* (Hrsg.):
Kostenrechnung im wissenschaftlichen Rechenzentrum - Das Göttinger Modell
1996

Nr. 44 *Plesser, Theo und Peter Wittenburg* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1996
1997

Nr. 45 *Koke, Hartmut und Engelbert Ziegler* (Hrsg.):
13. DV-Treffen der Max-Planck-Institute - 21.-22. November 1996 in Göttingen
1997

Nr. 46 **Jahresberichte 1994 bis 1996**
1997

Nr. 47 *Heuer, Konrad, Eberhard Mönkeberg und Ulrich Schwardmann*:
Server-Betrieb mit Standard-PC-Hardware unter freien UNIX-Betriebssystemen
1998

Nr. 48 *Haan, Oswald* (Hrsg.):
Göttinger Informatik Kolloquium - Vorträge aus den Jahren 1996/97
1998

Nr. 49 *Koke, Hartmut und Engelbert Ziegler* (Hrsg.):
IT-Infrastruktur im wissenschaftlichen Umfeld - 14. DV-Treffen der Max-Planck-Institute, 20. - 21. November 1997 in Göttingen
1998

Nr. 50 *Gerling, Rainer W.* (Hrsg.):
Datenschutz und neue Medien - Datenschutzschulung am 25./26. Mai 1998
1998

Nr. 51 *Plesser, Theo und Peter Wittenburg* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1997
1998

Nr. 52 *Heinzel, Stefan und Theo Plesser* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1998
1999

Nr. 53 *Kaspar, Friedbert und Hans-Ulrich Zimmermann* (Hrsg.):
Internet- und Intranet-Technologien in der wissenschaftlichen Datenverarbeitung - 15. DV-Treffen der Max-Planck-Institute, 18. - 20. November 1998 in Göttingen
1999

Nr. 54 *Plesser, Theo und Helmut Hayd* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 1999
2000

Nr. 55 *Kaspar, Friedbert und Hans-Ulrich Zimmermann* (Hrsg.):
Neue Technologien zur Nutzung von Netzdiensten - 16. DV-Treffen der Max-Planck-Institute, 17. - 19. November 1999 in Göttingen
2000

Nr. 56 *Plesser, Theo und Helmut Hayd* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 2000
2001

Nr. 57 *Hayd, Helmut und Rainer Kleinrensing* (Hrsg.):
17. und 18. DV-Treffen der Max-Planck-Institute
22. - 24. November 2000 in Göttingen
21. - 23. November 2001 in Göttingen
2002

Nr. 58 *Macho, Volker und Theo Plesser* (Hrsg.):
Forschung und wissenschaftliches Rechnen - Beiträge zum Heinz-Billing-Preis 2001
2003

Nr. 59 *Suchodoletz, Dirk von*:
Effizienter Betrieb großer Rechnerpools - Implementierung am Beispiel des Studierendennetzes an der Universität Göttingen
2003