17 Schlanke Endgeräte

Unternehmen und andere Einrichtungen klagen, dass sie ihre Server- und Anwender-PCs nicht genug auslasten, nur aufwändig verwalten und sichern und zu viel Energie für Speisung und Kühlung verbrauchen.

Gegen diese Probleme helfen u. a. folgende technische Entwicklungen:

- Der Ausbau der weltweiten IP-Netze erlaubt es, IT-Strukturen an beliebigen Orten unserer Welt einzurichten und an beliebigen anderen Orten zu nutzen. Dieses Überall-Computing heißt auch Ubiquitous Computing.
- Durch Virtualisierung (s. Kapitel 15) kann man Server und Desktops von der Hardware entkoppeln, Rechner besser ausnutzen und zentral betreuen. Damit kommen wir zur grünen Seite der Entwicklung.
- Verwaltungssoftware kann virtuelle Server und Anwender-PCs automatisch bereitstellen.
- Um überall auf virtuelle Anwender-PCs zugreifen zu können, braucht man Übertragungsprotokolle. Lesen Sie dazu in Kapitel 16.
- Schließlich brauchen Anwender noch schlanke Geräte, mit denen sie auf zentral betriebene Betriebssysteme und Anwendungen zugreifen. Diese stellen zwar weniger Rechenleistung zur Verfügung, sparen aber im Vergleich zu gewöhnlichen PCs Energie. Das ist Thema dieses leicht grün angehauchten Kapitels.

17.1 Konzepte für schlanke Endgeräte

Lesen Sie hier über Lösungen mit schlanken Endgeräten, die entweder nur Arbeitssitzungen auf Terminal-Servern und exportierten Desktops darstellen oder selbst lokal Anwendungen ausführen:

- PCs für Terminal-Sitzungen (Abschnitt 17.2),
- PCs als X-Terminals (Abschnitt 17.3),
- Flash-ROM-Terminals (Abschnitt 17.4),
- Grundlagen des Netzwerk-Bootens (Abschnitt 17.5),
- LTSP-Terminals (Abschnitt 17.6) und
- OpenSLX-Diskless Workstations (Abschnitt 17.7),

17.2 PCs für Terminal-Sitzungen

Exportierte Desktop- und Terminaldienst-Sitzungen von Windows- oder Linux-Terminalservern oder Desktops sowie von Mainframes kann man auf traditionell mit Festplatten ausgestatteten PCs mit Windows- oder Linux/Unix-Betriebssystemen nutzen, indem man diese mit Clients für die verschiedenen Transportprotokolle wie

- Remote Desktop Protocol (RDP) von Microsoft für Terminaldienste und entfernte Desktops,
- Independent Computing Architecture (ICA) von Citrix für Microsoft Server,
- X.11 für Linux/Unix,
- Nomachine NX von Nomachine für Microsoft-Terminaldienste und für Linux/Unix,
- 3270 für IBM-Mainframes
- oder Webclients
- ausrüstet.

Damit behalten Mitarbeiter ihre bisherigen Endgeräte und merken es vielleicht gar nicht, wenn sie exportierte Desktops oder serverbasierte Anwendungen statt lokaler Programme nutzen.

Technische Herausforderungen wie das Einbinden lokaler Geräte (Drucker, Laufwerke, Messgeräte etc.) sind inzwischen weitgehend gelöst.

Bereits dieses Weiterverwenden fetter PCs als Endgeräte für zentralisierte Datenverarbeitung löst viele Sicherheits- und Administrationsprobleme. Weniger Strom verbrauchen dagegen die weiter unten genannten Lösungen.

17.3 PCs als X-Terminals

In vielen Organisationen und Kleinbüros gibt es noch unzählige ältere Pentium-Computer mit stabiler Hardware, die für die Anforderungen der aktuellen ressourcen-hungrigen Windows-Versionen zu behäbig sind und zu wenig Arbeitsspeicher bieten.

Viele dieser Geräte kann man noch viele Jahre lang hervorragend an Benutzer-arbeitsplätzen als X-Terminals einsetzen, um darauf serverbasierte Linux- und Windows-Anwendungen darzustellen. Wenn die eingebaute Grafikkarte weniger als 1024 * 786 Bildpunkte oder eine Farbtiefe von weniger als 16 Bit bietet, spendieren Sie den alten Pentiums vielleicht noch eine neue Grafikkarte, solange es noch Karten für den PCI- beziehungsweise AGP-Bus gibt.

Solche Geräte können Sie von ihrer lokalen Platte oder von einem CD-ROM-Laufwerk booten oder ohne lokale Laufwerke von Linux-Servern.

Im Kapitel 16 haben Sie gelesen, wie Sie von einem Linux-Rechner den X-Desktop exportieren. Lesen Sie hier, wie Sie die Festplatte des Uralt-Pentium-PCs vorbereiten müssen, um die bewährten Geräte als lokal bootende X-Terminals nutzen zu können:

- Auf dem Server konfigurieren Sie das X-System so, dass es Anfragen von anderen Rechnern zulässt (siehe Kapitel 16).
- Auf den Client-Rechnern installieren Sie ein minimales Linux-System mit X-System, welches eine Verbindung mit dem Linux-Server aufnehmen kann.

Bei der hier beschriebenen administratorfreundlichen Lösung arbeiten die Benutzer an Linux-Endgeräten, die nur Sitzungen auf dem Linux-Terminalserver darzustellen brauchen. Diese Linux-Endgeräte sollen von einem eigenen Laufwerk (Festplatte, Flash-Speicher oder CD-ROM) booten.

Um nur grafische Sitzungen darzustellen, sind die Anforderungen an so einen Linux-Client nicht sehr hoch. Er sollte aber über eine Grafikkarte verfügen, die eine Auflösung von 1024x768 Punkten bei 16 Bit Farbtiefe darstellen kann.

Auf dem Client-PC bzw. einem Muster eines Client-PCs sollten Sie eine minimale Version von Linux mit X und dem Paket `xdmsc` aus der Serie *System • X11* installieren. Die anderen Auswahlmöglichkeiten wie Büroanwendungen, Dokumentation und KDE sind nicht notwendig.

Tipp: Auf älteren Rechnern mit weniger als 64 MByte Hauptspeicher ist auch die Linux-Installation sehr langsam. Bauen Sie einfach die Festplatte dieses Computers in ein aktuelles Gerät und installieren Sie dort. Nach der Installation bauen Sie die Festplatte zurück und vervollständigen die Konfiguration.

Das hier beschriebene Minimalsystem hat einen Umfang von ca. 350 MByte. Je nach Boot-Gerät können Sie über den Grundbedarf hinaus vorhandenen Speicher verschieden verwenden:

- wenn Sie es von einer CD-ROM booten, ist noch Platz für kleine Linux-Anwendungen und
- wenn Sie von einer Festplatte booten, steht der ganze darüber hinausgehende Platz für eine Swap-Partition oder Linux-Anwendungen zur Verfügung.

Bei X-Terminal-Clients müssen Sie damit rechnen, dass Benutzer diese einfach ausschalten, statt sie geordnet herunterzufahren. Statt Festplatten mit dem relativ robusten Dateisystem *reiserfs* können Sie inzwischen immer preisgünstigere Flash-Speicher verwenden.

Sobald X lokal läuft, können Sie Ihren Client mit wenigen Schritten mit YaST zum X-Terminal machen.

Die notwendigen Schalter finden Sie im YaST-Kontrollzentrum unter *System* • *Editor für /etc/sysconfig Dateien* unter *Desktop* • *XDMSC*. Hier ändern Sie die folgenden Werte:

```
RX_RHOST="192.168.1.2"     die IP-Adresse Ihres X-Servers
RX_XDMCP="query"
START_RX="yes"
```

Um danach das System zu testen, rufen Sie die grafische Oberfläche auf:

```
/etc/init.d/rx  tty7
```

Damit starten Sie die grafische Oberfläche und binden sie an die Konsole 7 (`tty7`).

In ein etwas nüchternes Anmeldefenster können Sie bzw. Ihre Benutzer dann Ihren Benutzernamen und Ihr Kennwort eingeben. Wenn alles klappt, startet nach dieser Anmeldung die grafische Oberfläche mit KDE, die Programme laufen auf dem Linux-Server und Sie oder Ihre Benutzer arbeiten am Linux-Client.

Damit das Anmeldefenster beim Login weiterer Benutzer immer wieder erneut starten kann, müssen Sie noch die Datei `/etc/inittab` anpassen.

```
#
# /etc/inittab
#
# Copyright (c) 1996-2002 SuSE Linux AG, Nuernberg, Germany.  All
rights reserve
d.
#
# Author: Florian La Roche <feedback@suse.de>, 1996
#
# This is the main configuration file of /etc/init, which
# is executed by the kernel on startup. It describes what
# scripts are used for the different run-levels.
#
# All scripts for runlevel changes are in /etc/init.d/.
#
# This file may be modified by SuSEconfig unless CHECK_INITTAB
# in /etc/sysconfig/suseconfig is set to "no"
#

# The default runlevel is defined here
id:5:initdefault:

........ (gekürzt)

# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
```

```
# The "id" field  MUST be the same as the last
# characters of the device (after "tty").
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
#
#S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102

#
#  Note: Do not use tty7 in runlevel 3, this virtual line
#  is occupied by the programm xdm.
#

#  This is for the package xdmsc, after installing and
#  and configuration you should remove the comment character
#  from the following line:
#7:3:respawn:+/etc/init.d/rx tty7


# modem getty.
# mo:235:respawn:/usr/sbin/mgetty -s 38400 modem

# fax getty (hylafax)
# mo:35:respawn:/usr/lib/fax/faxgetty /dev/modem

# vbox (voice box) getty
# I6:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI6
# I7:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI7

# end of /etc/inittab
```

Durch Entfernen des Kommentarzeichens # vor der im Listing fett hervorgehobenen Zeile aktivieren Sie den Eintrag für die siebte Konsole `tty7`. Diese Änderungen treten in Kraft, sobald Sie

```
telinit q
```

aufrufen. Danach sollte das grafische Anmeldefenster starten.

Wenn Sie Ihre Arbeitsplatz-PCs statt von lauten lokalen Festplatten über geräuschlose Boot-PROMs von Linux-Boot-Servern starten wollen, lesen Sie bitte im übernächsten Unterkapitel weiter. Lernen Sie zunächst Industrielösungen kennen, die – ähnlich wie hier für minimale Linux-Systeme beschrieben – Ihr Betriebssystem und Kommunikations-Clients für X.11, RDP, ICA; Browser etc. aus einem Flash-Speicher laden.

17.4 Flash-ROM-Terminals

Flash-ROM-Terminals halten ihr Betriebssystem in lokalen Halbleiterspeichern statt auf Festplatten. Je nachdem, ob man Settop-Boxen für Media-Anwendungen mitzählt oder nicht, führen im Markt hier Linux-Geräte vor Windows-Geräten oder umgekehrt. Inzwischen bietet die Terminal-Industrie eine immer größere Auswahl von so genannten Thin Clients, die überwiegend von lokalen Speichern, aber auch von Boot-Servern (siehe nächster Abschnitt) oder von benachbarten Terminals booten können.

Microsoft hat Windows-Terminals mit all seinen Varianten, angefangen mit Windows CE bis zu den aktuellsten .NET- und Embedded-Windows-Varianten, an Terminal-Hersteller lizenziert. Linux-Lösungen stecken u. a. weltweit in Millionen von Settop-Boxen für den Internetzugang und in Receivern für terrestrisches und Satellitenfernsehen beziehungsweise für Hörfunk.

Die Terminal-Hersteller sind bemüht, ihre Endgeräte immer mehr durch leistungsfähigere Hardware (schnellere Prozessoren, mehr Speicher, mehr Schnittstellen) und Software (mehr Client-Programme für mehr Übertragungsprotokolle, lokale Büroanwendungen) aufzuwerten, wodurch die Übergänge zu PCs fließender werden.

Solche Terminals sind einfach einzurichten und mit Fernverwaltungslösungen zentral administrierbar. Das Setup vieler solcher Terminals fragt nur nach allgemeinverständlichen Daten wie Adresse des Servers, IP-Adresse oder DHCP und Endbenutzer-Sprache. Übertroffen wird ihre Flexibiltät nur noch von Geräten mit Boot-PROM, die ganz darauf verzichten, lokal Programme vorzuhalten.

Browser-Appliances sind Endgeräte, die nur einen Browser als Benutzerschnittstelle für alle möglichen Programme bieten. Appliances können sich in Settop-Boxen für Fernseher, in Bildschirmtelefone, in Kiosksysteme und noch viele andere Geräteformen hüllen. Linux scheint sich immer mehr als Betriebssystem dieser Einfachst-Geräte durchzusetzen, zumal für das Open-Source-Betriebssystem Linux und den Open-Source-Browser Mozilla keine Lizenzgebühren anfallen. In vielen dieser Appliances steckt die hier beschriebene Flash-ROM-Technik.

Statt von einem lokalen Flash-Speicher kann man schlanke Endgeräte auch über das Netz von einem benachbarten Client oder von einem Server booten. Grundlagen dazu lesen Sie im nächsten Abschnitt. Dann geht es um vollständige Lösungen wie das Linux-Terminalserver-Projekt und Diskless Workstations.

17.5 Grundlagen des Netzwerk-Bootens

Dieser Abschnitt geht kurz auf grundlegende Konzepte und Protokolle für das Booten von Netz-PCs ein. Durchgesetzt hat sich hier die proprietäre BIOS-Erweiterung von Intel, das Preboot-eXtension-Environment (PXE). PXE ist inzwischen auf fast jedem

Mainboard oder Laptop verfügbar. Sie schalten diesen Service im BIOS Ihrer Clients ein und ändern die Boot-Reihenfolge.

Tipp: Hier spielt die im BIOS definierte Bootreihenfolge eine entscheidende Rolle: Steht die LAN-Option vielleicht erst an dritter Stelle nach dem Booten von CD-ROM oder Festplatte, bekommen Administratoren PXE gar nicht zu sehen, wenn der PC von einem anderen Datenträger erfolgreich starten konnte. Deshalb sollten Sie die LAN-Boot-Option stets an erster Stelle eintragen.

Bei Unklarheiten konsultieren Sie das Handbuch Ihrer Maschine.

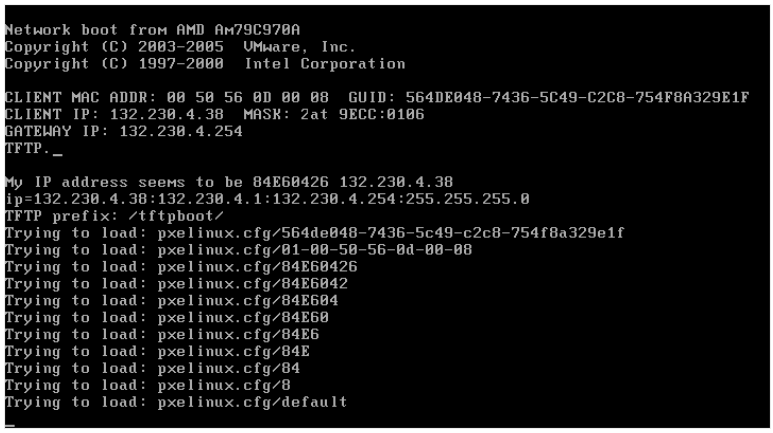

Abbildung 17.1: *Das typische Aussehen eines PXE-Netzstarters*

Auf der Client-Seite sind Sie nun fertig. Beide in späteren Abschnitten dieses Kapitels vorgestellten Netboot-Konzepte basieren auf DHCP (vgl. hierzu Kapitel 5) und TFTP zur Beschickung ihrer Clients mit den notwendigen Boot-Dateien und IP-Daten.

Ein Server für Netboot-Clients benötigt deshalb mindestens die beiden Dienste DHCP und TFTP. Der erste läuft vielleicht schon, der zweite ist eher selten bereits installiert. Für OpenSLX Stateless X Stations und LTSP benötigen Sie mindestens noch NFS. Letzteres ist Gegenstand von Kapitel 8.

Für Experimente mit OpenSLX-Clients überprüfen Sie am besten zuerst, ob die notwendigen Dienste bereits auf Ihrem Server installiert sind:

```
server:/root # rpm -q nfs-server atftp dhcp-server syslinux
package nfs-utils is not installed
package atftp is not installed
package dhcp-server is not installed
package syslinux is not installed
```

Sollte dieses nicht der Fall sein, installieren Sie diese mittels YaST2 nach.

17.5.1 DHCP- und TFTP-Service für Netboot-Clients

Bitte lesen Sie zunächst die allgemeine Einführung zu DHCP im Kapitel 5. In den meisten Fällen genügt es, für DHCP zwei Dateien anzupassen. Welche Clients der DHCP-Server mit welchen Informationen bedient, legt die Datei `/etc/dhcpd.conf` fest. Für einen ersten Test genügen die folgenden Eintragungen:

```
ddns-update-style none;
subnet 192.168.1.0 netmask 255.255.255.0 {
server-identifier 192.168.1.2;
filename pxelinux.0;
option routers 192.168.1.2;
host openslx-test {
  hardware ethernet 00:11:09:1D:64:E3;
  fixed-address 192.168.1.101;}
}
```

In der `/etc/sysconfig/dhcpd` bestimmen Sie, dass Ihr DHCP-Server auf dem passenden Interface, im Beispiel `eth0`, lauscht und zur Sicherheit in einer `CHROOT`-Umgebung läuft:

```
DHCPD_INTERFACE="eth0"
DHCPD_RUN_CHROOTED="yes"
[ ... ]
```

Die weiteren Einstellungen können Sie einfach übernehmen.

```
rcdhcpd start
```

wertet diese Datei beim Start des Dienstes aus.

Bei dem Advanced Trivial File Transfer Protocol (ATFTPD) müssen Sie noch weniger einrichten. Hier passen Sie die Datei `/etc/sysconfig/atftpd` durch den Eintrag

```
ATFTPD_OPTIONS="--daemon "
ATFTPD_USE_INETD="no"
ATFTPD_DIRECTORY="/tftpboot"
```

an. Dadurch dürfen die Clients beim Start dieses Dienstes unterhalb des eingetragenen Pfads Dateien vom Server laden. Damit der Client auch weiß, wo er suchen muss, teilt ihm der DHCP-Server dies vorher mit.

Diese Information tragen Sie in der Konfigurationsdatei des DHCP-Servers in der Option `filename` ein. PXE kann große Kernel von mehreren Megabyte Umfang nicht sofort laden. Für dieses Problem enthält das Syslinux-Paket von Peter Anwin eine PXE-Bootoption. PXE startet dabei `pxelinux.0`, ein kleines Binärprogramm, das seinerseits alle weiteren Schritte veranlasst. Dieses haben Sie mit dem Syslinux-Paket bereits installiert und finden es unterhalb von `/usr/share/syslinux`. Damit der ATFTPD diese Datei auch ausliefert, kopieren Sie `pxelinux.0` in das vorher von Ihnen angelegte Verzeichnis `/tftpboot`. Nach allen Änderungen, denken Sie bitte daran die Dienste mit `rcdhcpd restart` und `rcatftpd restart` zu reinitialisieren.

Tipp: Wenn Sie die Dienste nicht neu starten, registrieren diese Ihre vorgenommenen Änderungen nicht.

17.5.2 Linux-Terminals mit Boot-Prom

Eine besonders sparsame Form von Linux-Endgeräten verzichtet ebenfalls auf Fest-Wechselplatten, Boot-CDs oder Flash-Speicher und bootet Linux übers lokale Netz von Boot-Servern. Hierfür gibt es kommerzielle Lösungen beispielsweise von Bootix (`http://www.bootix.com`) und zahlreichen freien Linux-Projekten, z. B. vom Linux-Terminal-Server-Projekt (LTSP). Bei Boot-PROM-Lösungen veranlasst ein sehr kleines Programm im Boot-PROM der Netzwerkkarte in mehreren Schritten über einen Dialog mit einem Boot-Server – hier im Buch einem Linux-Boot-Server –, das Betriebssystem Linux von eben diesem Server zu laden. Die Grundidee und verschiedene Versionen dieser Technologie beschreiben die folgenden Abschnitte.

Sollen die Diskless-Endgeräte nur als Terminals arbeiten, reicht schon ein Pentium PC ab 16 MB RAM mit passender Netzwerkkarte mit PXE/Boot-PROM. Zwar verbringen Systemverwalter zunächst mehr Zeit mit dem Einrichten, können aber später alle Installationen und Änderungen zentral auf dem Boot-Server pflegen. Während auch Endanwender ohne Computerkenntnisse die zuvor erwähnten Lösungen mit Flash-ROMs zum Laufen bringen, müssen hier erst Systemverwalter serverseitig fleißig sein. Sie haben zum Glück kaum mehr zu tun, um 100 Diskless PCs zu betreiben, als um eine Handvoll herkömmlicher PCs einzurichten. Wenn alles läuft, ist es genauso für Endanwender geeignet wie Flash-ROM-Lösungen. Wer 486er PCs ansonsten entsorgen müsste oder welche geschenkt bekommt, zaubert mit weniger als 20 Euro Materialkosten funktionsfähige Endgeräte ohne Festplatte und Diskettenlaufwerk. Zu den 20 Euro kommen allerdings eventuell noch Kosten für neue Grafik- und Soundkarten hinzu.

17.6 LTSP-Terminals

Das LTSP-Projekt um Jim Mc Quillan ist ein seit Jahren bewährtes Thin-Client-Projekt, das über eine sehr starke Community verfügt, die die Lösung sehr liebevoll pflegt, erweitert und dokumentiert. Auch der Online-Suport ist sehr nett.

Dieser Teil des Kapitels beschreibt, wie Sie plattenlose PCs von einem Boot-Server starten, um sie dann ggf. an Terminalservern oder Mainframes als Terminal zu betreiben. Solche Endgeräte sollten Anwender von fortgeschrittenen Linux-Benutzern einrichten lassen. Die Installationsarbeit setzt u. a. Grundkenntnisse in NFS (Network File System) voraus, wie sie das Kapitel 8 dieses Buchs vermittelt.

Die aktuelle Version 5 des Projektes ist gegenüber den bisherigen Versionen sehr stark überarbeitet worden, so dass viele ältere Dokumentationen zu LTSP kaum noch helfen. Bei der aktuellen Version ist LTSP sehr stark in die jeweilige Distribution integriert, also

auch in OpenSUSE 11.0. Die starke Integration macht sich auch darin bemerkbar, dass die Software nun nicht mehr von den Servern des LTSP-Projektes bezogen wird, sondern von OpenSUSE-Servern. Die Einstiegsseite findet sich unter `http://en.opensuse.org/LTSP`.

Das OpenSUSE-Projekt geht mit der LTSP-Integration noch einen Schritt weiter als andere Distributionen. Es integriert zusätzlich sein KIWI-Projekt. Das KIWI-Projekt erzeugt komplette Images von lauffähigen Linux-Systemen. Das KIWI-Image wird dann von den LTSP-Clients gebootet. Damit kann auf dem Client-System ebenfalls OpenSUSE 11.0 laufen, ohne dass eine aufwändige Installation notwendig wäre. Bei den vorherigen LTSP-Versionen musste man das von LTSP gelieferte Linux booten.

Da das KIWI-Projekt sehr intensiv an seiner Software arbeitet, darf man sich nicht wundern, dass die Dokumentation noch nicht so perfekt ist, wie man es von LTSP bisher gewohnt ist.

17.6.1 Überblick

Im Laufe dieses Abschnitts lesen Sie Schritt für Schritt, wie Sie X-Terminals (Clients) einrichten. Sobald Sie die X-Terminals eingerichtet haben, starten diese in folgenden Schritten:

- Das Boot-ROM initialisiert die Netzwerkkarte.
- Das Boot-ROM sendet eine `Bootp`-Anfrage an den DHCP-Server.
- Der DHCP-Server beantwortet die Anfrage und teilt dem Client eine IP-Adresse zu. Außerdem teilt er dem Client die Lage und den Pfad des zu ladenden Linux-Kernels mit.
- Der Client sendet eine TFTP-Anfrage an den Boot-Server. (TFTP: Trivial File Transport Protocol).
- Der TFTP-Server beantwortet die Anfrage und sendet dem Client den Linux-Kernel.
- Der Client bootet jetzt Linux. Der Linux-Kernel sendet eine `Bootp`-Anfrage an den Server.
- Der Server beantwortet diese Anfrage. Mit diesen Daten konfiguriert der Linux-Kernel des Clients sein Netzwerk-Interface und setzt den Rechnernamen.
- Der Linux-Client versucht, sein *root*-Filesystem per NFS zu mounten.
- Der NFS-Server exportiert das angeforderte Verzeichnis an den Client.
- Der `init`-Prozess auf dem Client startet.
- Auf dem Client startet das Betriebssystem mit grafischer Oberfläche.
- Benutzer können sich auf dem Rechner einloggen, sofern sie auf dem Server über einen gültigen Account verfügen. Ihnen steht jetzt ihre X-Windows-Oberfläche zur Verfügung, und sie können damit auf dem lokalen Linux-System oder auf anderen Linux-Rechnern arbeiten.

Die beiden Projekte haben die für das Client-System notwendige Software und einige Installationsskripte zu einem leicht nutzbaren Softwarepaket zusammengestellt.

17.6.2 Vorbereitungen für LTSP

Für LTSP 5 müssen Sie die Netzwerkkarte, über die die Client-Rechner angebunden werden sollen, mit einer feste IP-Adresse konfigurieren.

Auf dem Linux-Server benötigt man für Terminals à la LTSP einen DHCP-Server, einen TFTP-Server, einen NFS-Server sowie ggf. einen XDM-Server. Wenn noch keine DHCP-Server und TFTP-Server eingerichtet sind, holt die LTSP-Installation dies nach.

17.6.3 LTSP Software

Zur Installation muss das ursprüngliche Installationsmedium, meist die DVD, ebenfalls eingelegt sein, da einige zusätzliche Pakete auch von diesem Medium installiert werden. Für die Grundinstallation gibt es mehrere Möglichkeiten: Das reicht von einen *1-Click Install* ausgehend von der Website bis zum Download eines `.iso`-Images mit den notwendigen Paketen. Über das Image installieren Sie ältere Quellen als über das aktuelle Repository.

Lesen Sie hier, wie Sie LTSP an der Konsole mit dem Programm `zypper` installieren.

Der erste Schritt fügt die Repositories `ltsp` und `tools` hinzu.

```
boss:~ # zypper ar
http://download.opensuse.org/repositories/server:ltsp/openSUSE_11.0
ltsp
Füge Repository 'ltsp' hinzu [fertig]
Repository 'ltsp' erfolgreich hinzugefügt
Aktiviert: Ja
Automatisch auffrischen: Nein
URI:
http://download.opensuse.org/repositories/server:ltsp/openSUSE_11.0

boss:~ # zypper ar
http://download.opensuse.org/repositories/openSUSE:/Tools/openSUSE_11
.0 tools
Füge Repository 'tools' hinzu [fertig]
Repository 'tools' erfolgreich hinzugefügt
Aktiviert: Ja
Automatisch auffrischen: Nein
URI:
http://download.opensuse.org/repositories/openSUSE:/Tools/openSUSE_11
.0
```

Wenn dieser Schritt erfolgreich verlaufen ist, kann die Installation des Pakets `kiwi-desc-ltsp` starten. Dabei muss auch das Installationsmedium zur Verfügung stehen, von dem aus der Server installiert wurde, da auch davon Pakete abgerufen werden.

```
boss:~ # zypper in kiwi-desc-ltsp
Lade Metadaten von Repository 'openSUSE-11.0-Updates' herunter
[fertig]
Erzeuge Zwischenspeicher für Repository 'openSUSE-11.0-Updates'
[fertig]
Möchten Sie diesem Schlüssel 85753AA5EEFEFDE9, openSUSE:Tools OBS
Project <openSUSE:Tools@build.opensuse.or                    g>,
Fingerabdruck OA031153E2F5E9DB7151OD8C85753AA5EEFEFDE9 vertrauen?
[ja/NEIN]: ja
Schlüssel 85753AA5EEFEFDE9 zu den vertrauenswürdigen Schlüsseln
hinzufügen? [ja/NEIN]: ja
Erzeuge Zwischenspeicher für Repository 'tools' [fertig]
Möchten Sie diesem Schlüssel 663A267061E1CF11, server:ltsp OBS
Project <server:ltsp@build.opensuse.org>, Fi
ngerabdruck 6C29A88EC9DA271129783D56663A267061E1CF11 vertrauen?
[ja/NEIN]: ja
Schlüssel 663A267061E1CF11 zu den vertrauenswürdigen Schlüsseln
hinzufügen? [ja/NEIN]: ja
Erzeuge Zwischenspeicher für Repository 'ltsp' [fertig]
Lese installierte Pakete...

Die folgenden NEUEN Pakete werden installiert:
  easy-ltsp kiwi-desc-ltsp ltsp-server ltspfs nbd kiwi-tools kiwi-
pxeboot
kiwi-desc-usbboot kiwi-desc-netboot kiwi-desc-isoboot kiwi tftp
syslinux squashfs
rpm-python perl-XML-SAX perl-XML-NamespaceSupport perl-XML-LibXML-
Common
perl-XML-LibXML perl-Config-IniFiles mtools libedit0 smart

Gesamtgröße des Herunterladens: 7,7 M. Nach der Operation werden
zusätzlich 14,2 M genutzt.
Fortfahren? [JA/nein]: ja
```

Nachdem Sie hier *Ja* eingegeben haben, beginnen der Download und die Installation der notwendigen Pakete von der Internetquelle und der Installations-DVD.

Das Laden und Installieren der ca. 30 Pakete kann etwas Zeit in Anspruch nehmen.

17.6.4 Konfigurieren der LTSP-Software

Nach dem Laden der Pakete konfiguriert das Skript `kiwi-ltsp-setup` den Server und verändert dessen installiertes System.

Bevor man das Installationsskript aufruft, sollte man die Voreinstellungen für die Installation in der Datei `/etc/sysconfig/kiwi-ltsp` prüfen:

```
## Path: System/Kiwi-ltsp
## Description: kiwi-ltsp setup - Part of KIWI-LTSP as created by
CyberOrg
```

```
# Copyright (c) 2007 Cyberorg
# This program is free software; you can redistribute it and/or
modify it under
# the terms of the GNU General Public License as published by the
Free Software
# Foundation; either version 2 of the License, or (at your option)
any later
# version.
#
…
## Type:        integer(0:1)
## Default:     0
#Set LTSP_DEBUG "0" logs to file and "1" to log KIWI activity to
screen instead of log file
LTSP_DEBUG="1"

## Type:        list(NFS,NBD)
## Default:     NBD
#Set the image type, NFS and NBD supported, NBD is default
IMAGETYPE="NBD"

## Type:        string
## Default:     /mnt/10.3
#Set the installation source path or URL
SUSE_INSTALL_SOURCE="/mnt/11.0"

## Type:        string(yast2,rpm-dir)
## Default:     yast2
#Set the installation source type
SUSE_INSTALL_SOURCE_TYPE="yast2"

## Type:        list(10.3,11.0)
## Default:     11.0
#Set the openSUSE version to use for creating image
SUSE_VERSION="11.0"
```

Sehr wichtig ist hier die Zeile `SUSE_INSTALL_SOURCE`. An dem hier angegebenen Pfad muss die DVD oder ein `.iso`-Image der DVD der OpenSUSE-Installation liegen. Dazu hängt man entweder die DVD per Hand an diese Position ein, oder gibt hier den Pfad an, an dem die DVD eingehängt ist.

Als nächstes sollte man das Installationsskript einen Installations-Check durchführen lassen:

```
kiwi-ltsp-setup -p
```

Das Script sollte die folgenden Ausgaben erzeugen:

```
KIWI-LTSP: 2008-09-28-12:43:22: ====== Starting ======
KIWI-LTSP: 2008-09-28-12:43:22: Checking for mandatory packages.
KIWI-LTSP: 2008-09-28-12:43:23: All mandatory packages are installed
KIWI-LTSP: 2008-09-28-12:43:23: Backing up current configuration
```

```
/srv/tftpboot/KIWI/
/srv/tftpboot/KIWI/defkeymap.map
/srv/tftpboot/KIWI/lts.conf
/srv/tftpboot/KIWI/config.default
/srv/tftpboot/KIWI/clock
/srv/tftpboot/KIWI/defkeymap.name
/srv/tftpboot/KIWI/deployltspfiles
/srv/tftpboot/KIWI/ssh_known_hosts
/srv/tftpboot/KIWI/language
/srv/tftpboot/KIWI/keyboard
/etc/dhcpd.conf
/etc/sysconfig/dhcpd
/etc/exports
KIWI-LTSP: 2008-09-28-12:43:23: ====== Setup completed ======
```

Nun kann die eigentliche Konfiguration starten, die auch das KIWI-Image für die Client-Rechner erzeugt. Haben Sie dabei etwas Geduld.

Starten Sie diesen Schritt mit:

```
kiwi-ltsp-setup -p
```

Um das notwendige Image für den Client zu erzeugen, muss das Installationsskript einige Pakete vom Installationsmedium laden.

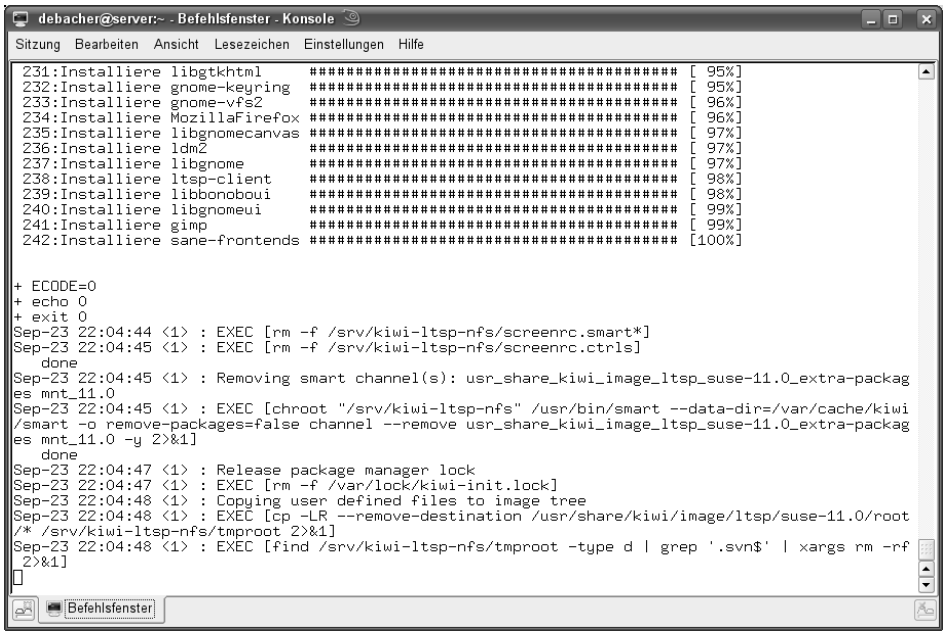

Abbildung 17.2: LTSP Setup

Wegen der hohen Zahl der notwendigen Pakete dauert das eine Weile.

Am Ende erzeugt das Skript das Image und das notwendige Dateisystem. Die letzten Zeilen der Ausgabe des Installationsscripts sollten deutlich machen, dass hier ein Image der Größe von 500 MB erzeugt wurde.

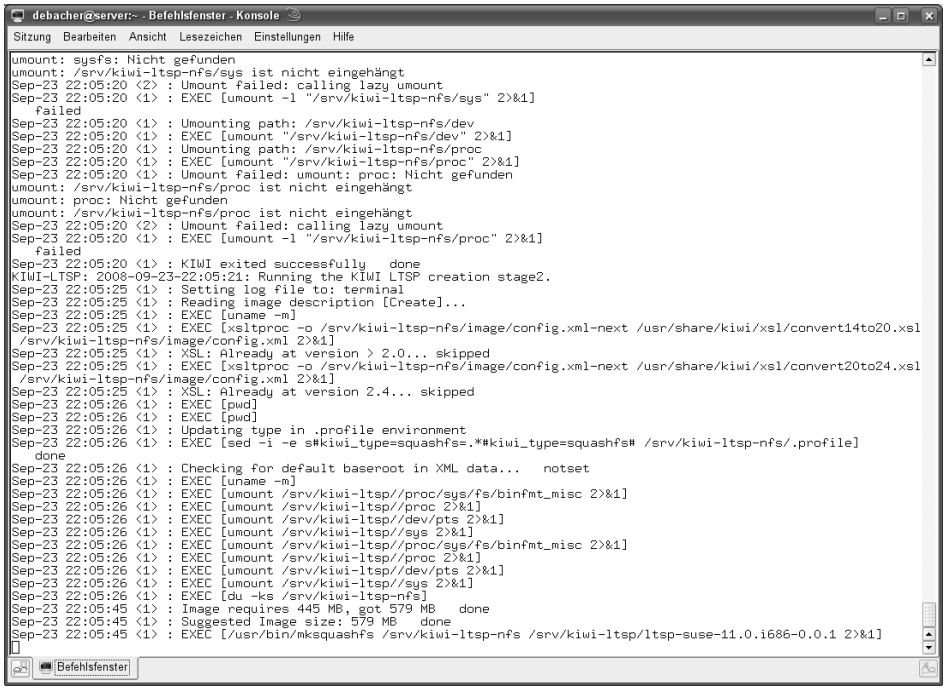

Abbildung 17.3: LTSP Ende des Setup

Kontrollieren sollte man, dass bei der Installation im Verzeichnis `/srv/tftpboot/boot/` die Dateien `initrd-ltsp` und `linux-ltsp` erzeugt wurden. Bei den Installationen der Redaktion war das nicht immer der Fall. Hier muss man dann eventuell die Dateien mittels `kiwi-ltsp-setup -n` explizit erzeugen.

Das Setup-Skript erzeugt neben dem Image auch passende Konfigurationsdateien, wobei es vorhandene Originale vorher sichert.

17.6.5 Nachbesserungen

Nachdem das Konfigurationsprogramm von LTSP seine Arbeit abgeschlossen hat, sollten Sie die wichtigsten Konfigurationsdateien prüfen.

Kontrollieren Sie auf alle Fälle die Datei `/etc/exports`:

```
/home   *.lokales-netz.de(rw,no_subtree_check)
/media/cdrom    *(ro,no_subtree_check)
/windows        www.linuxbu.ch(ro,no_subtree_check)
```

```
/srv/kiwi-ltsp-nfs
192.168.1.0/255.255.255.0(ro,no_root_squash,async,no_subtree_check)

/var/opt/ltsp/swapfiles
192.168.1.0/255.255.255.0(rw,no_root_squash,async,no_subtree_check)
```

An dieser Datei können Sie sehen, dass das Konfigurationsprogramm seine Einstellungen an die vorhandenen Einträge angehängt hat.

Die angegebenen IP-Adressen müssen unbedingt zu Ihrer Netzwerkkonfiguration passen, damit Ihre Client-Systeme auf die Freigaben zugreifen.

In der Datei `/etc/dhcpd.conf` passen Sie LTSP an Ihre eigenen Gegebenheiten wie Netzadressen und Hardware-Adressen spezieller Netzwerkkarten an.

Hier folgt ein Muster dieser `/etc/dhcpd.conf`:

```
# dhcpd.conf.template - Part of KIWI-LTSP as created by Cyberorg
#
# Copyright (c) 2007 Cyberorg
# This program is free software; you can redistribute it and/or
modify it under
# the terms of the GNU General Public License as published by the
Free Software
# Foundation; either version 2 of the License, or (at your option)
any later
# version.
#
# This program is distributed in the hope that it will be useful, but
WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
or FITNESS
# FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
more
# details.
#
# You should have received a copy of the GNU General Public License
along with
# this program; if not, write to the Free Software Foundation, Inc.,
59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#
# Authors: Cyberorg Cyberorg <cyberorg@cyberorg.info>
#         Magnus Boman <captain.magnus@gmail.com>
#
# Version       Date            Changes
# 0.1           2007-08-25      Initial release
```

```
option domain-name "lokales-netz.de";
option domain-name-servers 192.168.1.2;
option routers 192.168.1.2;
default-lease-time 14400;
ddns-update-style none;
next-server 192.168.1.2;
subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.50 192.168.1.100;
  default-lease-time 14400;
  max-lease-time 172800;
  filename "pxelinux.0";
}
```

Die Dateien, die der Client im ersten Schritt laden soll, liegen im Verzeichnis `/srv/tftpboot/pxelinux.cfg/` und werden vom Programm `/srv/tftpboot/pxelinux.0` geladen. Den Pfad `/srv/tftpboot/` muss man in der obigen Konfiguration weglassen, da er in der Konfiguration des TFTP-Servers voreingestellt ist.

Die Einträge in der Datei `/opt/ltsp/i386/etc/lts.conf` für die Workstation-Namen müssen mit den hier gesetzten Namen übereinstimmen. Rechner, die mit der Standardkonfiguration der `lts.conf` funktionieren, braucht man hier nicht einzutragen.

Bitte passen Sie danach die Datei `/srv/tftpboot/KIWI/lts.conf` an. Die Parameter sollten selbsterklärend sein. Die Datei `/srv/tftpboot/KIWI/lts.conf` ist die zentrale Konfigurations-Datei für die LTSP-Clients.

```
# This is the default lts.conf file for ltsp 5.
# For more information about valid options please see:
# /usr/share/doc/ltsp-client/examples/lts-parameters.txt.gz
# in the client environment

[default]
    SOUND=True
    LOCALDEV=True
    CONFIGURE_X=true
    SERVER=192.168.1.2
    SCREEN_07=ldm
    LOCAL_APPS=true
```

Um alle Änderungen wirksam zu machen, geben Sie nun ein:

```
init 1
```

und nach einem erneuten Einloggen als *root*:

```
init 5
```

17.6.6 Boot-Menü anpassen

Startet man den ersten Client mit dem LTSP-System, so erscheint auf dessen Bildschirm ein Boot-Menü.

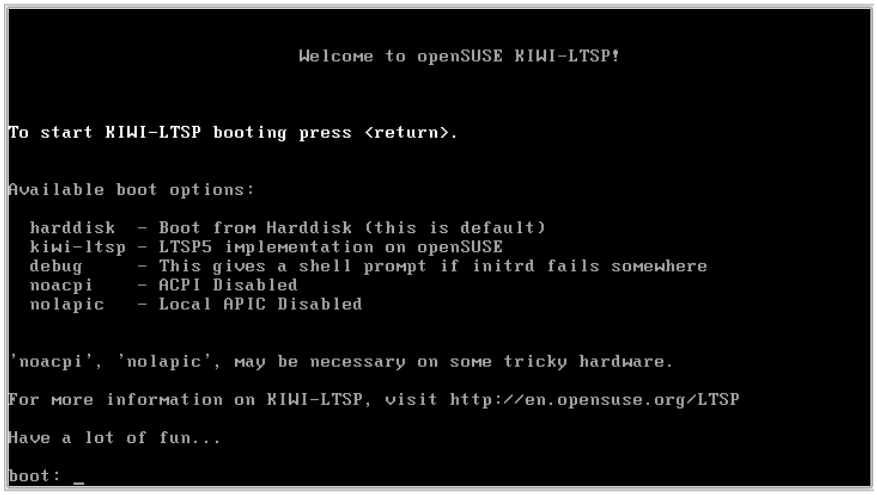

Abbildung 17.4: LTSP Boot-Menü

Will man dieses Menü anpassen, so muss man auf dem Server die Datei `/srv/tftpboot/pxelinux.cfg/default` bearbeiten.

```
implicit        1
display         message-ltsp
prompt          1
timeout         100

DEFAULT kiwi-ltsp

LABEL kiwi-ltsp
        kernel boot/linux-ltsp
        append initrd=boot/initrd-ltsp vga=791 splash=silent showopts
        IPAPPEND 2

LABEL debug
        kernel boot/linux-ltsp
        append initrd=boot/initrd-ltsp vga=791 acpi=off kiwidebug=1
        IPAPPEND 2

LABEL noacpi
        kernel boot/linux-ltsp
        append initrd=boot/initrd-ltsp vga=791 acpi=off
        IPAPPEND 2
```

```
LABEL nolapic
        kernel boot/linux
        append initrd=boot/initrd-ltsp vga=791 nolapic
        IPAPPEND 2

LABEL Local-Boot
        localboot 0
```

Der einfache Aufbau der Datei erinnert an die Konfiguration des Bootmanagers *Grub*. Für jeden der fünf Einträge im Bootmenü findet sich hier ein Abschnitt. Der Abschnitt `kiwi-ltsp` startet automatisch, wenn der Benutzer nicht innerhalb von ca. 10 Sekunden einen anderen Eintrag ausgewählt hat.

Um diese Boot-Menüs für jeden Rechner individuell anzulegen, muss man im Verzeichnis `/srv/tftpboot/pxelinux.cfg/` jeweils eine Konfigurationsdatei anlegen, deren Dateiname der MAC-Adresse des Rechners entspricht. Hat der Rechner die MAC-Adresse `00:0c:29:86:0c:ce`, so müsste diese Datei `000C29860CCE` heißen.

Hat man mehrere ähnliche Rechner, so sind in der Regel auch ihre MAC-Adressen ähnlich. Sie unterscheiden sich nur im hinteren Teil. Legt man eine Konfigurationsdatei mit dem Namen `000C29` an, so würde diese von allen Rechnern geladen werden, deren MAC-Adresse mit `00:0c:29` beginnt und für die es keine individuellere Konfigurationsdatei gibt.

Damit kann man die Boot-Einstellungen der Rechner sehr individuell konfigurieren.

17.6.7 Testen mit VMware

Am einfachsten fällt Ihnen das Testen, wenn auf Ihrem Server VMware zur Verfügung steht (siehe Kapitel 15). Die Software erlaubt auch das Booten per PXE.

Legen Sie also einfach einen neuen virtuellen Rechner an und booten diesen, ohne ihm eine Diskette oder eine CD/DVD anzubieten. Dann startet dieser virtuelle Rechner seinen PXE-Code.

Der Vorteil des Testens mit VMware liegt darin, dass Sie Probleme relativ leicht erkennen können, da alles auf dem gleichen Rechner abläuft.

Nach kurzer Zeit sollten Sie die Anmeldemaske Ihres Servers zu sehen bekommen.

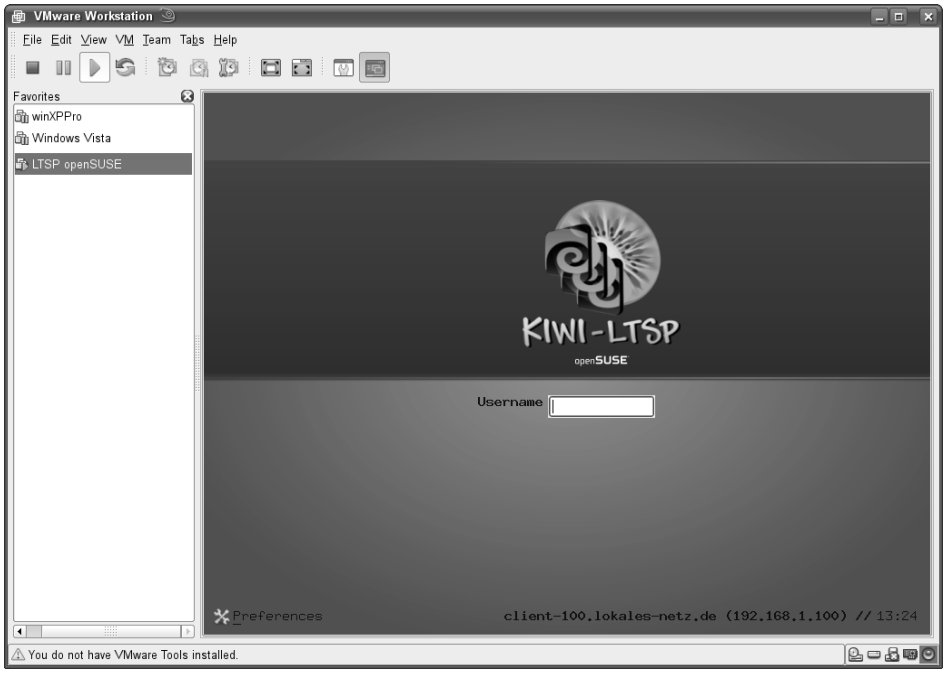

Abbildung 17.5: LTSP-Client in VMware

Tests der Redaktion mit VMware funktionierten immer sehr gut.

17.7 OpenSLX – Diskless Workstations

Statt plattenlose PCs nur als Remote-Terminals einzurichten, können Sie mit der GPL-Software OpenSLX vollständig lauffähige Linux-Arbeitsplatzrechner aufsetzen. Solche festplattenlosen Linux-Workstations sind einfach zu installieren und leicht zu administrieren.

OpenSLX ermöglicht auch große Installationen mit sehr vielen Clients. Durch Virtualisierungslösungen wie VMplayer können Sie auf einem OpenSXL-Client weitere Betriebssysteme effektiv verteilen und betreiben.

Das Paket OpenSLX ist, wie eigentlich jedes OpenSource-Paket, nicht fertig, sondern wird seit 1997 beständig weiter entwickelt. Aktuell ist derzeit die Version 5. Sie können diese von der Projekt-Homepage `www.openslx.org` als RPM- oder Debian-Paket herunterladen; für OpenSUSE 11.0 wählen Sie das RPM (die genaue Bezeichnung hängt von der jeweiligen Revision ab):

```
server:~ # wget http://openslx.org/common/packages/archive/openslx-
5.0.M-N.i586.rpm
server:~ # rpm -i openslx-5.0.M-N.i586.rpm
```

Neben dem Basispaket benötigen Sie noch die Pakete `perl-Clone`, `perl-Config-General` und nur für die Entwicklungsversionen `subversion`, die Sie am bequemsten mittels YaST hinzufügen.

Für aktuelle Bugfixes oder neue Features wie weitere unterstützte Distributionen für Ihre Clients können Sie jederzeit anonym per Versionsverwaltung auf die Quellen im Entwicklungsbaum zugreifen:

```
server:~ # svn co http://svn.OpenSLX.org/svn/openslx/openslx
...
server:~ # Checked out revision 2NNN.
slx-server:~ # cd openslx/trunk
slx-server:~/openslx/trunk # make install
```

Nach der Installation auf dem einen oder anderen Weg stehen Ihnen OpenSLX-Administrationskommandos zur Verfügung, die mit dem Präfix `slx` beginnen. Mit

```
server:~ # slxsettings set private-path='/var/opt/openslx' set
public-path='/'
setting private-path to '/var/opt/openslx'
setting public-path to '/'
```

bestimmen Sie, wo das Ausgangsdateisystem Ihrer SLX-Clients landet. Die Vorbereitungsdaten, Konfigurationen und die SLX-Datenbank, werden unterhalb von `/var/opt/openslx` abgelegt. Sie umfassen den Umfang einer Distribution, so dass Sie für das Verzeichnis mindestens 10 GByte vorsehen sollten. Der spätere NFS-Export erfolgt aus `/export` und die TFTP-Dateien kommen unterhalb `/tftpboot`. Wählten Sie einen anderen `public-path`, steht dessen Präfix vor den beiden letztgenannten Verzeichnissen. Die aktuelle Version Ihres OpenSLX-Toolsets liefert:

```
slx-server:~ # slxversion
5.0.0revNNNN
```

17.7.1 Funktionsweise

OpenSLX kümmert sich um die komplette Einrichtung von Netboot-Clients als Linux-Workstations. Dabei kann ein SLX-Server mehrere Systeme, die auch auf unterschiedlichen Linux-Distributionen basieren können, bereitstellen.

Für eine gute Übersichtlichkeit ist die Setup-Prozedur für die Netboot-Clients in mehrere Schritte aufgeteilt:

- *Stage0* bezeichnet ein laufendes Referenzsystem auf realer oder virtueller Hardware, von dem durch `rsync` ein Abzug auf den Server erzeugt wird. Dazu muss `rsync` auf dem System installiert sein. Dieses Stadium spielt lediglich für den Klonmodus eine Rolle.
- *Stage1* bezeichnet ein lauffähiges Clientsystem, welches unterhalb von

- `/var/opt/openslx/stage1/<name>` landet. Bei übereinstimmender Hardware-Architektur können Sie durch `slxos-setup shell <name>` direkt in dieses Verzeichnis hineinwechseln. Das System wird grundsätzlich eingerichtet und ggf. um bestimmte lokale Erweiterungen wie von der Distribution nicht angebotene externe Softwarepakete ergänzt. Ein *Stage1* können Sie durch direkte Installation aus den jeweiligen Paketquellen oder durch den Abzug von einem bestehenden System (Klon, *Stage0*) mit den SLX-Tools einrichten.

- *Stage2* – aus dem *Stage1* lassen sich verschiedene Client-Root-Filesysteme erzeugen. Dazu dupliziert `slxos-export` das *Stage1* und lässt dabei nicht benötigte Verzeichnisse und Dateien, beispielsweise aus dem Verzeichnis `/var`, weg. `Slxos-export` erzeugt hierzu ein neues Unterverzeichnis im NFS-Export oder legt für Netzwerk-Blockdevices ein *SquashFS* an. *SquashFS* ist ein komprimiertes, nur lesbares Dateisystem. Es hilft, den Netzwerkdatenverkehr zu reduzieren. Dazu gehören mindestens ein passender Kernel (üblicherweise direkt aus der verwendeten Distribution), mindestens ein zum Kernel passendes InitialRamFS und eine Client-Konfiguration. Das InitialRamFS enthält alle zum Mounten des Rootfilesystems notwendigen Tools und führt die Vorbereitungsschritte aus. Die Client-Konfiguration kann Dateien enthalten, die im *Stage3* an die passende Stelle entpackt werden.

- *Stage3* – Während *Stage0-2* administrative Vorgänge auf dem SLX-Server umfasst, die Sie nur wenige Male ausführen müssen, findet *Stage3* auf jedem einzelnen Client bei jedem Neustart innerhalb des InitialRamFS statt. Hier setzen die Shell-Skripten `init` (SLX-Init), `hwautocfg` und `servconfig` den Client-PC individuell auf. Sie legen alle wichtigen Konfigurationsdateien des Clients in `/mnt/etc` an oder modifizieren diese. Lokale Erweiterungen (für alle Clients, die ein bestimmtes InitRamFS verwenden) sind mit `preinit.local`, ganz am Anfang von SLX-Init, und `postinit.local` ziemlich am Ende von SLX-Init möglich. `postinit.local` lässt sich individuell per Client in einem Paket aus Konfigurationsdaten (*ConfTGZ*) zur Verfügung stellen. Dieses Paket wird in *Stage3* beispielsweise per TFTP gezogen und anschließend entpackt. Die Vorlage hierzu liegt in `/var/opt/openslx/config/<system>`.

- *Stage4* konfiguriert aus SLX-Sicht nur noch Kleinigkeiten. Der Client hat ein komplettes Root-Filesystem, alle Programme und Dateien und kann jetzt seine Tastatur einstellen und sein grafisches Login starten.

- Je nach Art der Clients können Sie auf einem Server eine oder mehrere verschiedene Systemexporte bereitstellen. So können Sie ein *Stage1*-System in mehreren Varianten wie NFS und gleichzeitig NBD/SquashFS an die Clients ausliefern oder sehr unterschiedliche *Stage1* mit derselben Technologie aus unterschiedlichen Exporten bedienen. Das ist beispielsweise für Tests sinnvoll, wenn man neben dem aktuell benutzten System schon das neue System parallel anbieten möchte.

17.7.2 Programmnamen und globale Einstellungen

Die Konfigurations- und Setup-Programme liefern bei Angabe der Option `--help` eine Kurzhilfe. Das Handbuch zum Kommando gibt die hinten angestellte Option `--man` aus.

Nach der Installation können Sie mit dem Programm `slxsettings` globale Grundeinstellungen wie Pfade an Ihre Vorstellungen anpassen. Ohne Parameter aufgerufen, zeigt Ihnen `slxsettings` die Liste der aktuellen Einstellungen. Die Hilfe `slxsettings --man` listet die für alle SLX-Kommandos gemeinsamen Optionen auf:

```
NAME
       slxsettings - OpenSLX-script to show & change local settings
SYNOPSIS
       slxsettings [options] [action ...]
...
       General Options
             --help                    brief help message
             --man                     full documentation
             --quiet                   do not print anything
             --version                 show version
```

Für die meisten SLX-Umgebungen sollten die Voreinstellungen passen. Wenn Sie aber beispielsweise unter `/space` eine große leere Plattenpartition eingehängt haben und diese für OpenSLX nutzen möchten, können Sie die *Stage1*-Installationen auf diese umleiten:

```
slx-server:~ # slxsettings --private-path="/space"
setting private-path to '/space'
```

Ebenso können Sie die Art der Datenbank zur Verwaltung der mit OpenSLX exportierten Systeme und Ihrer Clients ändern. Der Standardtyp ist *SQLite*. Je nach installierten Perl-Komponenten können Sie alternativ auch MySQL als Backend einsetzen.

```
slx-server:~ # slxsettings --db-type=SQLite
setting db-type to 'SQLite'
```

Den Datenbanktyp können Sie im laufenden Betrieb später nicht mehr einfach ändern. Nach den Grundeinstellungen können Sie sich nun den Stadien 0 bzw. 1 widmen.

17.7.3 Stage1 aus Netzwerk-Quellen oder Referenzmaschine

Je nach Linux-Distribution auf den Clients stehen Ihnen ein oder zwei Wege des *Stage1*-Setups zur Verfügung: Entweder Sie installieren ein ganz frisches System aus den Paketquellen oder Sie klonen eine schon fertig eingerichtete Maschine, beispielsweise Ihre OpenSUSE 11.0-Maschine. Beides übernimmt das Kommando `slxos-setup`, welches den Aufbau `slxos-setup kommando <parameter>` hat. Auskunft gibt das Subkommando list-`supported`:

```
slx-server:~ # slxos-setup list-supported
List of supported distros:
        suse-11.0               (clone)
        suse-11.0_x86_64        (clone,install)
        ubuntu-8.04             (clone)
        ...
```

Eine Neuinstallation nimmt `slxos-setup` mit dem Subkommando `install` vor, gefolgt von der gewünschten Distribution:

```
slx-server:~ # slxos-setup install suse-11.0
[ ... lots of yum output ... ]
Total download size: 69 k
Downloading Packages:
(1/2): nbd-2.8.7-14.i586. 100% |=========================|  45 kB
00:00
(2/2): squashfs-kmp-defau 100% |=========================|  24 kB
00:00
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: squashfs-kmp-default         ############################
[1/2]
  Installing: nbd                          ############################
[2/2]

Installed: nbd.i586 0:2.8.7-14 squashfs-kmp-default.i586
0:3.1_2.6.18.2_34-34
Complete!
Starting SuSEconfig, the SuSE Configuration Tool...
Running in full featured mode.
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.perl...
Executing /sbin/conf.d/SuSEconfig.permissions...
Finished.
Vendor-OS 'suse-11.0' installed succesfully.
Vendor-OS 'suse-11.0' has been added to DB (ID=1).
```

Je nach Bandbreite und Auslastung Ihres Netzwerks können Sie eine Kaffee- oder Mittagspause bis zum Abschluss dieses Schrittes einplanen. Jede so aufgesetzte Distribution können Sie später mit `slxos-setup update <distro>` auffrischen, was dann deutlich schneller als die Primärinstallation geht.

Wenn Sie schon ein fertig eingerichtetes System haben, weil Sie beispielsweise bestehende Fat-Clients in OpenSLX-Clients umwandeln wollen, kann ein Klon-Setup eine sinnvolle Option sein. Das Subkommando `clone` benötigt dazu die Ausgangsquelle. Um ein geklontes System von einer klassischen RPM-Installation unterscheiden zu können, geben Sie dem Klon am besten einen individuellen Namen. Alles, was Sie

hinter dem Distributionsnamen und der Version angeben, im Beispiel `-mysystem`, wird Bestandteil des Namens.

Rsync klont im Hintergrund, so dass Sie am Anfang nach dem *root*-Passwort der Referenzmaschine gefragt werden. Auf einigen Referenzsystemen müssen Sie daher vorher dafür sorgen, dass `root` sich per SSH anmelden darf. Für einen schnellen Test können Sie einfach Ihren OpenSUSE 11.0-Server klonen:

```
slx-server:~ # slxos-setup clone localhost:/ suse-11.0-mysystem
Cloning vendor-OS from '11.00.4.50:/' to '/space/stage1/suse-11.0-
mysystem'...
Password:
receiving file list ...
...
var/yp/
var/yp/nicknames
var/yp/binding/

sent 5765235 bytes  received 6044617832 bytes  5710602.23 bytes/sec
total size is 6025777602  speedup is 1.00
Vendor-OS 'suse-11.0-mysystem' has been cloned succesfully.
Vendor-OS 'suse-11.0-mysystem' has been added to DB (ID=2).
```

Das Subkommando `list-installed` zeigt Ihnen alle angelegten Stage1-Systeme an:

```
slx-server:~ # slxos-setup list-installed
List of installed vendor-OSes:
        suse-11.0-mysystem
        suse-11.0
```

17.7.4 Plugins – Modulare Erweiterungen des Basissetups

Die generelle Software-Architektur von OpenSLX zielt darauf, sämtliche über die Grundfunktionalität (die Bereitstellung und Verknüpfung der einzelnen Stages) hinausgehende Fähigkeiten des Systems mittels spezieller Erweiterungen, den *Plugins*, zu implementieren.

Die bisherigen Schritte genügten daher nur für ein bootfähiges Grundsystem, jedoch nicht für einen komfortablen grafischen Desktop. Diesen erstellen die beiden Plugins `xserver` und `desktop`. Das erste richtet X.org automatisch ein und bindet vielleicht gewünschte proprietäre OpenGL-Treiber von Nvidia oder ATI/AMD ein. Das zweite kümmert sich um das Setup des gewünschten Displaymanagers und der grafischen Default-Sitzung.

Das Kommando `slxos-plugin` installiert Plugins und deren vielleicht von den Voreinstellungen abweichende Einstellungen:

```
slx-server:~ # slxos-plugin install suse-11.0 desktop manager=kdm
Plugin desktop has been installed into vendor-OS 'suse-11.0'.
slx-server:~ # slxos-plugin install ubuntu-8.04-clone xserver

Plugin xserver has been installed into vendor-OS 'suse-11.0'.
```

Attribute, die wie beim `desktop`-Plugin die Einstellungen für den später laufenden Client beinhalten, schreibt `slxos-plugin` in die Datenbank. Dort können Sie diese jederzeit für einzelne Clients, ganze Gruppen oder OpenSUSE 11.0 ändern. Diese Feinheiten Ihrer Clients richten Sie mit dem im übernächsten Abschnitt vorgestellten Werkzeug `slxconfig` ein. Neben diesen Plugins stehen noch eine Reihe weiterer im Basispaket zur Verfügung:

- *Bootsplash* kann einen grafischen Startbildschirm anzeigen, der vor dem Erscheinen des grafischen Logins gezeigt wird. Hierzu setzt es *Splashy* ein, welches im Userspace läuft und somit das Patchen des Kernels überflüssig macht.
- Das Plugin `x11vnc` richtet den VNC-Server (siehe Kapitel 16) ein. Dieser erlaubt es, auf eine exakte Kopie des X.org-Bildschirms der lokalen Konsole zuzugreifen.
- Syslog richtet den Syslogger auf einem OpenSLX-Client ein.

Tipp: Vergessen Sie nach der Plugin-Installation nicht erneut zu exportieren und anschließend den `slxconfig-demuxer` laufen zu lassen.

17.7.5 Systeme exportieren

Nach diesen Vorarbeiten erzeugen Sie für Ihre Netboot-Clients ein Rootfile-System. `slxos-export` veranlasst dabei dem Übergang von der *Stage1*-Installation in *Stage2*-Exports. Rufen Sie `slxos-export` mit dem Parameter `list-installed` auf, präsentiert Ihnen dieser das identische Ergebnis wie `slxos-setup list-installed` und zeigt Ihnen damit exportierbare Installationen an:

```
slx-server:~ # slxos-export list-installed
List of installed vendor-OSes:
        suse-11.0-mysystem
```

Diese Systeme können Sie nun exportieren. Unterstützt sind bisher die Typen `nfs` und `nbd-squash`. Mit dem Unterkommando `export` legen Sie einen neuen Export an, beispielsweise:

```
slx-server:~ # slxos-export export suse-11.0 sqfs
```

Ein klassisches NFS-Root-Filesystem legen Sie durch einen ähnlichen Aufruf an:

```
slx-server:~ # slxos-export export suse-11.0-mysystem nfs
```

Anschließend können Sie sich durch `slxos-export list-exported` vergewissern, dass bis hierhin alles funktioniert hat.

Bis zu diesem Punkt haben Sie das Root-Filesystem (entweder das klassische NFS oder NBD mit SquashFS) vorbereitet und die Datenbank mit Basisdaten gefüllt. Die bisherigen Schritte erlauben Ihnen nach dem Abschluss der SLX-Einrichtung den Start Ihres festplattenlosen Clients bis zum Konsolen-Login. Möchten Sie Ihren Benutzern einen grafischen Desktop anbieten oder sogar weitere Systeme durch virtualisierte Maschinen anbieten, installieren Sie die im letzten Abschnitt eingeführten Plugins.

17.7.6 Die Datenbank und das Boot-Setup

Das Kommando `slxconfig-demuxer` stellt die jeweiligen Kernel mit dem dazu passenden InitialRamFS bereit und konfiguriert die Clients. Es arbeitet direkt mit den Einträgen der Datenbank und benötigt deshalb in den meisten Fällen keine Parameter. Mit den von OpenSLX erzeugten Basisdaten erreichen Sie bereits eine gültige Konfiguration, die für jeden anfragenden Client gilt. Feinheiten Ihrer Clients richten Sie mit dem Werkzeug `slxconfig` ein.

Sobald Sie mehr Systeme verwalten wollen oder die Clients unterschiedliche Einstellungen benutzen sollen, empfiehlt sich eine weitergehende Konfiguration per Datenbank. Mit der Datenbank, die `slxos-*` unterhalb von `/var/opt/openslx/db` oder in dem von Ihnen definierten Verzeichnis und Typ anlegt, unterhalten Sie sich mit dem Kommando `slxconfig`.

Die bisherigen Einträge in die Datenbank haben `slxos-setup` und `slxos-export` automatisch eingetragen:

```
slx-server:~ # slxconfig list-vendoros
List of the matching vendor-OSes:
        suse-10.1-mysystem
      suse-11.0
slx-server:~ # slxconfig list-export
List of the matching exports:
        suse-10.1-mysystem        (nfs)
   suse-11.0:sqfs-nbd        (sqfs-nbd)
```

Den letzten Schritt vollziehen Sie durch den Aufruf von `slxconfig-demuxer`. Dieses Kommando benötigt keine weiteren Subkommandos oder Parameter, da es alle Informationen direkt aus der Datenbank bezieht. dieser Schritt erledigt alle wesentlichen Operationen zum Abschluss der *Stage2*-Phase.

Abbildung 17.6: Erfolgreicher Start eines OpenSLX-Clients mit Bootsplash-Plugin

Der `slxconfig-demuxer` baut dabei die Verzeichnisse `/tftpboot` und `/tftpboot/client-config` bei jedem Durchlauf komplett neu auf. Bisherige Änderungen in diesem Bereich gehen verloren. Die Quellen des Demuxers sind vielfältig:

- Der Kernel und dessen Module stammen aus dem *Stage1*, dem jeweiligen Hersteller-OS, von dem ein Export erzeugt wurde.
- Die Basiskonfiguration, das `initramfs-setup`, für das erstellte InitRamFS, wird komplett aus der Datenbank generiert. Einträge ändern Sie mit `slxconfig`.
- Das Paket der Konfigurationsdaten stammt aus `/var/opt/openslx/config/<system-name>`. Der Systemname wird beim Eintrag in die Datenbank durch `slxos-export` automatisch generiert.

17.7.7 Zugriff auf die Datenbank

Das Werkzeug `slxconfig` stellt die allgemeine Benutzerschnittstelle zur Datenbank dar. Damit können Sie sich alle Einstellungen anzeigen lassen und diese ändern. So zeigt `slxconfig list-system` alle aktivierten, für Clients potenziell bereitgestellten SLX-Systeme. Die gesetzten Variablen der einzelnen Clients sehen Sie mit:

```
slx-server:~ # slxconfig --verbose list-system name="<system>"
```

Die meisten Einstellungen sind dabei nicht belegt und werden für das Erstellen des `machine-setup` vom Default-System `"<<<default>>>"` übernommen. Sie können sowohl Parameter Ihres Systems als auch des Defaultsystems ändern:

```
slx-server:~ # slxconfig change-system "<<<default>>>" country=de
system '<<<default>>>' has been successfully changed.
```

Diese Konfiguration lokalisiert alle Ihre angelegten Systeme für Deutsch. Sie müssen diesen Eintrag daher nicht für jedes System wiederholen.

Um die Heimatverzeichnisse der User auf dem klassischen NFSv3-Weg bereitzustellen, tragen Sie in die Datenbank ein:

```
slx-server:~ # slxconfig change-system suse-11.0-mysystem:sqfs-nbd
attr_automnt_dir=/home
attr_automnt_src="nfs://192.168.1.2/home"
```

Abbildung 17.7: Grafisches Login eines OpenSLX-Clients im Vmware-Player

Eine einfache Benutzerkonfiguration erhalten Sie, wenn Sie Ihre `/etc/passwd` und `/etc/shadow` in das Systemkonfigurations-Verzeichnis `/var/opt/openslx/config/suse-11.0-mystem/default/rootfs/etc` kopieren und anschließend den

`slxconfig-demuxer` erneut starten. Hier können Sie weitere Konfigurationsdateien ablegen, die Sie angepasst auf Ihren OpenSLX-Clients bereitstellen wollen.

Tipp: Vergessen Sie nach Anpassungen der Konfigurationsdatenbank oder Änderungen von Dateien im Systemkonfigurationsverzeichnis nicht, den `slxconfig-demuxer` neu zu starten.

17.7.8 System-Update und Ergänzungen

In direkt aus den Paketquellen installierten *Stage1*-Installationen können Sie Updates auf dem Server vornehmen lassen. Dieses löst der `update`-Parameter des `slxos-setup` aus:

```
slx-server:~ # slxos-setup update suse-11.0
Setting up Update Process
Setting up repositories
base_non-oss              100% |=========================|  951 B
00:00
base                      100% |=========================|  951 B
00:00
base_update               100% |=========================| 1.2 kB
00:00
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 548 kB
00:00
########################################################## 1340/1340
Resolving Dependencies
[ ... etliche weitere Ausgaben ... ]
Starting SuSEconfig, the SuSE Configuration Tool...
Running in full featured mode.
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.perl...
Executing /sbin/conf.d/SuSEconfig.permissions...
Finished.
Vendor-OS 'suse-11.0' updated succesfully.
```

Klonsysteme aktualisieren Sie auf der Referenzmaschine im sogenannten *Stage0*. Anschließend rufen Sie erneut `slxos-setup clone localhost:/ suse-11.0-mysystem` auf:

```
slx-server:~ # slxos-setup clone localhost:/ suse-11.0-mysystem
building file list ...
[ ... etliche weitere Ausgaben ... ]
sent 434920 bytes  received 121491994 bytes  375737.79 bytes/sec
```

```
total size is 8644862874  speedup is 70.90
Vendor-OS 'suse-10.1-mysystem' has been recloned succesfully.
No need to change vendor-OS 'suse-11.0-mysystem' in OpenSLX-database.
```

Abschließend meldet `slxos-setup` seinen Erfolg. Für Systemklone gibt es keinen speziellen Update-Parameter, da das Update außerhalb des Servers auf der Referenzmaschine erfolgt und erst dann auf den Server kopiert wird.