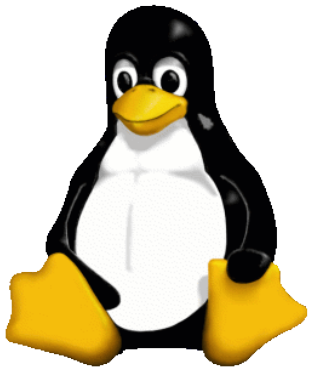
Implementierung eines skalierbaren Terminalservers für plattenlose Clients unter Linux



Projektarbeit Systemadministration

NTA FH Isny
12. Info

David Mayr

Inhalt

1 Einführung

Um viele einzelne Computer mit deren Betriebssystem und allen benötigten Anwendungen zu betreuen Bedarf es eines recht hohen administrativen Aufwands[1], sei es z.B. nur für das Update oder die Installation einer bestimmten Software auf allen Systemen. Um dem Abhilfe zu schaffen, gibt es schon recht lange[2] die Idee einen sehr leistungsfähigen Server ode Mainframe zu verwenden und viele "dumme" Terminals anzuschliessen, um mit diesen direkt auf dem Server zu arbeiten.
Mit Linux hat man nun die Möglichkeit, mit einfachen, und vor allem gegenüber großen Servern oder Mainframes wesentlich günstigeren Standard-PCs eine zentral administrierbare und in Grenzen skalierbare Umgebung einzurichten, ohne dass dabei jegliche Lizenzkosten wie bei vergleichbaren kommerziellen Lösungen[3] entstehen. Zusätzlich ist es bei ausreichender Bandbreite[4] an den Terminals sogar möglich, komplett auf Festplatten zu verzichten da alles auch zentral über einen Boot-Server bezogen werden kann.

2 Grundlagen / Theorie

In diesem Abschnitt möchte ich die der Projektarbeit zugrundeliegende Theorie erläutern, die sich im wesentlichen auf die folgenden drei Hauptgebiete konzentriert:
- Was ist ein Terminalserver und was versteht man unter Server Based Computing?
- Wie ist es möglich Computer ohne lokalen Massenspeicher (z.B. Festplatte) über das Netzwerk mit einem Betriebs- und Dateisystem zu versorgen?
- Was ist ein Cluster, und wie kann es helfen, die hohe Server-Last zu reduzieren?

2.1 Was ist Server Based Computing?

Beim Server Based Computing arbeiten viele Benutzer mit Ihren eigenen Clientsystemen – den Terminals – auf einem oder mehreren sehr leistungsstarken Terminal-Servern. Alle Anwendungen bzw. Prozesse die über die Terminals gestartet werden, werden auf dem Server ausgeführt. Nur Tastatur-, Maus- und Bildschirminformationen werden über das Netzwerk zwischen dem Terminal und dem Server übertragen – der Benutzer arbeitet, als würde er direkt am Server sitzen. Das Terminal muss lediglich netzwerkfähig sein und eine entsprechende Client-Anwendung starten, die dann Verbindung mit dem Server aufnimmt um mit der Authentifikation des Benutzers fortzufahren und ihm die gewünschte Anwendung auf dem Server zu starten. So ist es bei den Clients auch möglich, sehr leistungsschwache Hardware einzusetzen.
Je nach Art der Arbeitsumgebung und Sitzungsart auf dem Server – grundlegend unterscheiden lassen sich textbasierte und grafische – und der Anzahl der Terminals, sind unterschiedliche Netzwerk-Bandbreiten für einen störungsfreien Betrieb erforderlich.
Ein wesentlicher Vorteil dieser Technologie ist, dass der Administrationsaufwand durch die zentrale Bereitstellung aller Anwendungen und Daten auch bei einer sehr hohen Benutzer- und Terminal-Anzahl auf ein Minimum reduziert wird. Allerdings steht der Vorteil, dass die Terminals mit sehr schwacher Hardware[5] ausgestattet sein können gegenüber dem Nachteil, dass dadurch ein oder sogar mehrere leisungsstarke Terminal-Server bereitstellt werden müssen.

2.2 Was sind Plattenlose Clients?

1 ~~Spätesten wenn die Anzahl der Com~~puter 20-50 übersteigt – erst recht, wenn es sich um mehrere hundert Computer handelt.
2 früher allerdings nur mit Textbasierten Terminals.
3 z.B. den Windows Terminal Services oder Citrix MetaFrame
4 Mindestens 100 Mbit sind empfehlenswert, für die Server-Anbindung wäre eine Gbit-Verbindung zu einem entsprechenden Switch empfehlenswert – spätestens wenn die Anzahl der grafischen X-Terminals 50 übersteigt.
5 es reichen beispielsweise ein Pentium60, min. 16MB, 2MB Grafikkarte und je nach verwendetem Protokoll von 64 KBit bis 100 Mbit Netzwerk für eine grafische(!) Sitzung.

Mit einem wie oben beschriebenen Terminal-Server werden also alle Anwendungen[6] auf dem Server ausgeführt und sind somit zentral administrierbar. Alles, was bis hier noch direkt auf den Terminals verwalten werden muss, ist das Dateisystem und der zu bootende Betriebssystem-Kernel. Sofern man sich in einem zuverlässigen und schnellen LAN befindet ist es mit Linux möglich, den Kernel vom Netzwerk zu laden und das root-Dateisystem über NFS[7] einzubinden. Somit werden alle Terminals vollständig zentral administrierbar.
Auf dem Boot-Server, der zugleich auch Termial-Server sein kann, sind dazu mindestens DHCP[8]-, TFTP[9]- und NFS-Server nötig. Ausserdem muss ein angepasster Kernel für die Terminals bereitgestellt werden – doch dazu später mehr.

Der Bootvorgang eines solchen plattenlosen Clients gliedert sich in die folgenden Schritte:

- Computer startet und das BIOS leitet die Boot-Phase ein.
- Es wird ein Netzwerk-Bootloader gestartet – es gibt im wesentlichen zwei Standards[10], die aber recht ähnlich arbeiten.
- Der Netzwerk-Bootloader startet. Er besteht aus den folgenden Komponenten und benötigt nur einige wenige KB Speicherplatz:
  - Treiber für die jeweilige Netzwerkkarte.
  - DHCP-Client, der vom Server eine IP-Konfiguration und weitere Parameter wie z.B. die Adresse des TFTP-Server und den Namen der Kernel-Datei empfängt.
  - TFTP-Client, um den zu bootenden Kernel[11] vom Server übers Netzwerk zu übertragen.
- Starten des Linux-Kernels, initialisieren der Geräte und Nachladen von Kernel-Modulen – z.B. für die Netzwerkkartenunterstützung - aus der initialen RAM-Disk[12].
- Mounten des root-Dateisystemes per NFS vom in der DHCP-Antwort enthaltenen Server.
- Starten von init, das sich um den Start weiterer nötiger grundlegender Dienste kümmert.
- Einleiten der Terminal-Sitzung auf dem Server.

Es ist mit einigen Kniffen und geschickten Shell-Scripts möglich, für alle Terminals ein einziges root-Dateisystem im schreibgeschützten Modus zu exportieren. Die Terminals erstellen sich beim Booten dann eine kleine RAM-Disk, die sie für temporäre Dateien und sonstige Schreibzugriffe nutzen. Ausserdem kann die Hardware[13] in den meisten Fällen automatisch konfiguriert werden. Das hat den enormen Vorteil, dass man unabhängig von der Anzahl der Terminals immer konstant gleich viel Speicherplatz[14] für die root-Dateisysteme der Termials benötigt.

2.3 Was ist ein Cluster?

Da alle Anwendungsprozesse auf dem Terminal-Server ausgeführt werden, erhöht sich dessen Rechenlast mit jedem zusätzlichen Anwender und dessen Aktivitäten am Terminal. Damit der Server unter der Last bei sehr vielen Terminals nicht zusammenbricht, müssen entweder sehr leistungsstarke und teure Server angeschafft, oder auf anderem Wege die Rechenlast bewältigt werden. Da heutzutage günstige Standard-PCs schon enorme Leistungen bringen, bietet es sich an diese bei den Terminal-Clients ungenutzte Rechen-Kapazität sinnvoll zu verwerten.
Ein Cluster ist algemein ein Verbund von Rechnern, die sich die Rechenlast teilen. Bei vielen Cluster-Implementationen müssen die Anwendungen speziell für die jeweilige Technik programmiert worden sein, vor allem für z.B. wissenschaftliche Anwendungen.
Für Linux gibt es zur transparenten[15] Last- bzw. Prozessverteilung über ein Netzwerk das OpenSource-

6 natürlich ausser dem Betriebssystem und seinen Werkzeugen, sowie der Anwendung die für die Verbindung mit dem Terminal-Server sorgt.
7 NFS, das Network File System ermöglicht es, Dateisysteme übers Netzwerk zu exportieren so dass mehrere Clients auf einen zentralen Datenbestand zugreifen können
8 DHCP ist das Dynamic Host Configuration Protocol, um Clients im Netzwerk die eine entsprechende Anfrage stellen eine IP-Konfiguration zuzuweisen. Dabei schickt der Client seine MAC-Adresse (Ethernet) per Broadcast an alle Netzwerkteilnehmer und der DHCP-Sever reagiert je nach Konfiguration. Neben der reinen IP-Konfiguration können auch noch weitere Daten wie z.B. spezielle Boot-Parameter per DHCP übermittelt werden.
9 TFTP ist das "Trivial FTP" – ein stark abgespecktes File Transfer Protocol, mit dem Ziel möglichst einfache und Speicherplatzsparende Clients für die grundlegendste Dateiübertragung in einem IP-Netzwerk zu ermöglichen.
10 Das PXE (Pre-boot eXecution Environment) ist auf vielen moderneren Mainboards v.a. mit onboard-LAN schon im BIOS integriert. Die andere Alternative nennt sich Etherboot und kann z.B. auf eine Diskette, aber auch auf ein EPROM als Boot-ROM auf einer Netzwerkkarte geschrieben werden, um von einem Server zu booten.
11 Und eine eventuell verwendete initiale RAM-Disk (initrd), mehr dazu später.
12 Die initiale RAM-Disk ist ein gzip-komprimiertes Dateisystem-Image (meist ext2), das ein minimales root-Dateisystem mit Kernel-Modulen und den entsprechenden Lade-Tools für Geräte enthält, die geladen werden müssen bevor das eigentliche root-Dateisystem gemountet werden kann (z.B. IDE-, Netzwerk- oder Dateisystem-Treiber). Die initrd wird in der gleichen Art wie das Kernelimage dem Bootloader übergeben und von diesem geladen.
13 Vor allem die Grafikkarte und der Monitor für den grafischen Modus
14 Bei diesem Projekt benötigt das komplette root-Dateisystem für alle (!) Terminals weniger als 200 MB
15 Anwendungen müssen nicht speziell angepasst werden.

Projekt openMosix. Es handelt sich hierbei um einen sehr umfangreichen Kernel-Patch und eine Handvoll Programme im Userspace. Ein so gepatchter und neu compilierter openMosix-Linux-Kernel ist in der Lage, Prozesse[16] über das Netzwerk selbständig bei Bedarf an andere Cluster- / openMosix-Knoten zu verteilen. Die Knoten, welche so Prozesse eines anderen Knoten abgearbeitet haben, senden das Ergebnis Ihrer Arbeit anschliessend wieder zum ursprünglichen Rechner zurück der dann damit weiterarbeitet.
Wenn nun alle Terminals und der Server in so einem openMosix-Clusterverbund arbeiten, können die Terminals, welche sonst nur die wenig rechenintensive Aufgabe der Anzeige und Netzwerkkommunikation haben, den Server entlasten indem sie Prozesse für diesen abarbeiten.

2.4 Vor- und Nachteile

Der entscheidende Vorteil dieser Lösung ist vor allem der stark verminderte Adminstrationsaufwand bei nur durch Leistungsfähigkeit der Hardware[17] und Bandbreite der Netzwerkinfrastruktur beschränkter Terminal-Anzahl, da alles auf einem Server verwaltet werden kann – sowohl Daten, Konfiguration als auch Systemdateien. Ausserdem bringt jedes Terminal Leistung mit, die es dem gesamten Cluster zur Verfügung stellt und damit dort eingesetzt werden kann, wo die Leistung benötigt wird – i.d.R. auf dem Terminal-Server.
Ein Nachteil dieser Technik ist die starke Abhängigkeit von der Zuverlässigkeit des Servers – sollte dieser seine Dienste nicht mehr anbieten, sind alle Terminals für die Dauer der Störung nicht verwendbar. Eine Abhilfe wäre zum Beispiel eine möglichst redundante Auslegung aller kritischen Bauteile und die Eleminierung aller "Single Points of failure". Zusätzlich könnte man noch einen Hot-Standby Backup-Server bereithalten, der im Fehlerfall die Arbeit eines ausgefallenen Servers übernehmen könnte.

3 Umsetzung

Bei der im folgenden beschriebenen praktischen Umsetzung und den Konfigurations-Beispielen gehe ich davon aus, dass es einen Server mit der IP-Adresse 10.0.2.250/255.255.254.0 (Netzwerk 10.0.2.0/Broadcast 10.0.3.255) gibt, auf dem alle benötigten Dienste ausgeführt werden. Denkbar wäre bei sehr großen Installationen auch einen oder mehrere Server pro Dienst (DHCP/TFTP/NFS) einzurichten.

3.1 Server-Grundsystem

Prinzipiell wäre jede aktuelle Linux-Distribution als Grundlage für den Server geeignet. Da die Gesamt-Leistung des Systems jedoch in besonderem Maße von der Geschwindigkeit des Servers abhängt, wurde Gentoo Linux gewählt. Das besondere an dieser sog. Meta-Distribution ist, dass einzelne Software-Packete nicht als Binär-Programme von einem Datenträger installiert, sondern durch ein ausgeklügeltes System mit dem Namen "portage" im Quellcode heruntergaladen und für den jeweiligen CPU-Typ – z.B. Athlon-XP - optimiert compiliert werden. Dabei werden alle eventuellen Packet-Abhängikeiten[18] selbständig aufgelöst. Somit erhält man bis zu 30% schnelleren Maschinen-Code gegenüber Distributionen deren Programme für alle CPU-Typen ab dem 486er oder Pentium compiliert wurden – allerdings kann der Code in der Regel dann auf keinem anderen CPU-Typ mehr ausgeführt werden[19]. Eine sehr detailierte Anleitung zur Installation findet man auf der Gentoo-Homepage unter http://gentoo.org.

3.2 Boot-Server

Der Boot-Server versorgt die Terminals mit IP-Daten, einem Betriebssystem-Kernelimage und dem root-Dateisystem, so dass diese keinerlei Massenspeicher wie z.B. Festplatten brauchen.

3.2.1 DHCP

Der DHCP-Server ist zentraler Bestandteil des Boot-Servers, hier werden für jedes Terminal – identifiziert durch seine MAC-Adresse – IP-Daten und weitere Konfigurationsparameter bereitgestellt. Bei den

16 Vorzugsweise Prozesse mit wenig IO-Bedarf und hohen CPU-Anforderungen
17 Wo zur Not auch einfach weitere Termial-Server zur Verfügung gestellt werden könnten, sofern das durch die Lastverteilung im Cluster überhaupt nötig sein sollte (erst bei sehr vielen Terminals)
18 z.B. benötigt ein Videoplayer auch alle Bibliotheken die zum Abspielen seiner Video- und Audio-Formate nötig sind.
19 d.h. alles müsste komplett neu compiliert werden – kann je nach gewünschter Softwareausstattung und Leistung der vorhandenen Hardware bis zu mehreren Tagen dauern.

meisten Distributionen findet man im Verzeichniss `/etc` oder `/etc/dhcp/` die Datei `dhcpd.conf`, die für den Termial- bzw. hier den Boot-Server wie folgt aussehen könnte:

```
allow               bootp;
allow               booting;
default-lease-time  1209600;
max-lease-time      31536000;
ddns-update-style   none;
option domain-name  "nta.net";
next-server         10.0.2.250;                         # TFTP-Server für PXE-Clients
server-name         10.0.2.250;                         # TFTP-Server für Etherboot-Clients

subnet 10.0.2.0 netmask 255.255.254.0 {

     range                          10.0.2.1  10.0.2.99               # für dyn. Adressvergabe
     option domain-name-servers     10.0.2.250, 80.81.6.17;           # DNS-Server
     option routers                 10.0.2.250;                       # Standard-Gateway

     if substring (option vendor-class-identifier, 0, 9) = "Etherboot" {
            filename "linux-kernel.etherboot-image";                  # für Etherboot-Terminals
     } else {
            filename "pxelinux.0";                                    # für PXE-Terminals
     }

     host termial-001 {  hardware ethernet  00:0D:60:FF:39:B3;  fixed-address      10.0.2.100;  }
     host termial-002 {  hardware ethernet  00:50:9c:15:33:ca;  fixed-address      10.0.2.101;  }
     host termial-003 {  hardware ethernet  00:d0:09:e9:98:26;  fixed-address      10.0.2.102;  }
     host termial-004 {  hardware ethernet  00:00:e8:6c:8c:f7;  fixed-address      10.0.2.103;  }
     ...
}
```

Details zu den einzelnen Konfigurationsparameter lassen sich mit **`man dhcpd.conf`** und **`man dhcp-options`** nachlesen. Im wesentlichen werden hier Host-Namen, IP-Adressen und die Dateinamen der Kernel-Images – die anschliessend per TFTP geladen werden sollen - pro Netzwerk/Subnetz vergeben.

3.2.2 TFTP

Das Trivial File Transfer Protocol wird verwendet, um das Kernelimage übers Netzwerk zum Terminal zu übertragen. In den meisten Fällen wird der TFTP-Daemon vom Internet-Superserver `inetd` bzw. seinem Nachfolger `xinetd` auf Anforderung an Port 69 gestartet – es ist aber auch möglich den TFTP-Daemon ständig laufen zu lassen, was eine schnellere Reaktionszeit beim booten der Terminals bringt.
Unabhängig davon muss dem TFTP-Daemon über die Konfigurationsdatei[20] mitgeteilt werden, in welchem Verzeichniss die Kernel-Images zu finden sind – hier wurde `/tftpboot` gewählt.

3.2.3 NFS

Auf dem Server muss nun noch für die Terminals ein root-Dateisystem per NFS exportiert werden. Dazu muss zuerst ein root-Dateisystem vorhanden sein, das exportiert werden kann. Im Prinzip könnte man dafür einfach eine bestehende Minimal-Installation[21] seiner bevorzugten Linux-Distribution verwenden und dann entsprechende init- und boot-Scripte[22] schreiben, die dafür sorgen dass

1. alle für die vorhandene Netzwerkkarte erforderlichen Kernel-Module[23] aus der initialen RAM-Disk geladen werden,
2. ein DHCP-Client gestartet wird, der den Hostnamen setzt und u.a. die IP-Adresse des NFS-Servers in Erfahrung bringt, um dann
3. von diesem das root-Dateisystem im schreibgeschützten Modus zu mounten. Dann muss noch
4. eine RAM-Disk für temporäre Dateien und sonstige Schreibzugriffe[24] angelegt und mit den entsprechenden Dateien verlinkt werden, und
5. letztendlich init die Kontrolle übergeben wird, das sich um den gewohnten Boot-Prozess kümmert

20 Abhängig von Distribution und Verwendung von (x)inetd – unter Gentoo-Linux mit dem atftpd ist die Konfiguration unter /etc/conf.d/atftp
21 mit zumindest (für den grafischen Betrieb) einem vollständigen X-Server und den benötigten NFS- und sonstigen System-Tools.
22 genaugenommen nur ein Script (/linuxrc) das zuerst anstelle von init aus einer initialen RAM-Disk startet und erst später wieder die Boot-Kontrolle an init übergibt.
23 Anhand eines PCI-Bus Scan wird der Name des Kernel-Moduls (mithilfe einer Zuordnungstabelle) i.d.R automatisch richtig erkannt.
24 z.B. legt der DHCP-Client mehrere Dateien an, und

und am Ende den X-Server startet der eine Verbindung zum Termial-Server aufnimmt und einen grafischen Login anbietet.

Zum Glück muss das Rad in der OpenSource-Welt nicht ständig neu erfunden werden – um die Implementierung eines solchen Scripts und vorallem des root-Dateisystems der Terminals hat sich schon das Linux Terminal Server Projekt (http://ltsp.org) gekümmert. Das root-Dateisystem wird dabei unter `/opt/ltsp/i386` abgelegt, und dieses muss auch in der Datei `/etc/exports` per NFS exportiert werden:

```
/opt/ltsp                10.0.2.0/255.255.254.0(ro,no_root_squash,sync)
```

3.3 Terminal-Server

Für rein textbasierte Terminals wäre es mit z.B. einem SSH- oder Telnet-Server an dem sich die Benutzer der Terminals anmelden können auf der Server-Seite schon getan. Für die grafische Anmeldung der Terminals gibt es mehrere Möglichkeiten, wovon ich hier im folgenden zwei erwähnen möchte.

3.3.1 XDMCP / X11 / X-Window

Für die Bereitstellung eines grafischen Logins für die Terminals auf dem Server reicht der Standard X11-Server, bzw. der X-Display Manager XDM der schon von Haus aus Netzwerkfähig ist. Dafür wird das Protokoll XDMCP[25] verwendet, das die Bildinformation zum Terminal (im X-Sprachgebrauch der X-Server) und die Eingabegeräte-Events zum Terminal-Server (X-Client) überträgt. Auf dem Server muss dazu zum einen in der Datei `/etc/X11/xdm/Xaccess` der Zugang zum XDMCP freigegeben werden, und ausserdem z.B. der KDE-Anmeldungsmanager KDM für die grafischen Logins konfiguriert[26] werden.

3.3.2 NX / freeNX

Da das schon recht alte XDMCP leider noch keine Komprimierung verwendet und auch die Netzwerkbandbreite nicht optimal ausnutzt, wird vor allem bei Grafik-intensiven Anwendungen auf mehreren Terminals – z.B. aufwändige Bildschirmschoner - auch ein 100MBit-Netzwerk schnell zu langsam. Alternativ wäre es deshalb sinnvoll, das mit sehr geringem Netzwerkoverhead verbundene NX-Protokoll von der Firma nomachine[27] zu verwenden. Damit wäre es sogar möglich, über eine Internet-Einwahlverbindung mit nur 56 Kbit/s eine grafische Desktop-Sitzung ruckelfrei zu steuern.
Um das zu realisieren, müssen auf dem Server alle freien NX-Komponenten und das Server-Script des freeNX-Projektes installiert, und im root-Dateisystem der Terminals der freie NX-Client mit den erforderlichen Bibliotheken installiert werden. Auch muss dafür gesorgt werden, dass jetzt nicht mehr eine X11 Remote-Sitzung gestartet wird, sondern der X-Server und anschliessend der NX-Client.

3.3.3 Einschränkungen

Da alle Prozesse – auch eventuelle Druck-Spooler oder Programme für die Audio-Ausgabe – sowie alle direkte Zugriffe auf die Hardware auf dem Server ausgeführt werden, sind mit diesem Umfeld gewisse Einschränkungen verbunden. Das Problem der Druckeransteuerung lässt sich mithilfe von Netzwerkdruckern lösen, die Audio-Ausgabe kann mit dem X11-Protokoll zum jeweiligen Terminal übertragen werden. Dass nicht direkt auf die Hardware wie zum Beispiel den CD-ROMs an den Terminals zugegriffen werden kann, muss in grösseren Installtionen gar nicht unbedingt ein Nachteil sein – oft sollen die Benutzer gar keine Daten von extern unkontrolliert in das System bringen können.

3.4 openMosix Cluster

3.4.1 Kernel patchen und compilieren

Um alle Terminals und den Server Clusterfähig zu machen, müssen zwei gepatchte Linux-Kernel erstellt

25 Das X-Display Manager Control Protocol
26 In Gentoo-Linux mit KDE-3.3.2 in der Datei /usr/kde/3.3/share/config/kdm/kdmrc
27 Kernbestandteile unter der GNU General Public License – nähere Informationen dazu unter http://www.nomachine.org. Der NX-Client ist für viele Betriebssysteme erhältlich, unter anderem neben Linux auch für Windows

und compilert werden – einer optimiert für den Server, und einer speziell für die Terminals. Dazu muss zuerst der aktuelle openMosix-Patch von deren Homepage unter http://openmosix.sourceforge.net heruntergeladen werden und nach Anleitung ein installierter Kernel-Quellbaum gepatcht werden. Für den Terminal-Kernel sind gegenüber dem Server z.B. keine IDE-Treiber nötig.

3.4.1 Installation des Terminal-Kernels

Nachdem der Kernel mit den Modulen für die Terminals compiliert ist, muss dieser noch in ein über das Netzwerk bootfähiges Binärformat mithilfe des Programms `mknbi-elf` überführt[28] werden und eine initrd mit den nötigen Netzwerkkartentreibermodulen erstellt werden.

4 Fazit

Mit dieser Technik ist es möglich, viele grafische Terminals für einen breiten Anwendungsbereich mit minimalem Aufwand zu erstellen und zentral administrieren. Durch entsprechend optimerte Linux-Kernel ist es möglich, ein Terminal innerhalb von ca. 20 sekunden komplett vom Netzwerk zu booten und dem Benutzer eine grafische Anmeldemaske anzubieten.

5 Quellenangaben

Internet:

http://ltsp.org	Projekt-Homepage des Linux Terminal Server Project
http://gentoo.org	Homepage von Gentoo-Linux
http://openmosix.sf.net	Seite des openMosix Cluster-Projekts für Linux
http://google.org/linux	Linux-Suche mit Google

Literatur:

c't Magazin (Heise) 1/2005 S. 178ff – "Office aus dem Netz" (SBC vereinfacht die Softwarepflege)

28 Mit PXE ist das nicht nötig, nur bei Verwendung von Etherboot. Für PXE sind noch weitere Bootloader-Komponenten nötig, auf die ich hier nicht näher eingehen werden, da dies den Rahmen des Projektes bei weitem sprengen würde.