Netzwerkboot über Weitverkehrsnetze

Ansatz zur zentralen Steuerung und Verwaltung von Betriebssystemen über das Internet

Sebastian Schmelzer*, Dirk von Suchodoletz*, Gerhard Schneider*

{sebastian.schmelzer,dirk.von.suchodoletz,gerhard.schneider}@rz.uni-freiburg.de

*Albert-Ludwigs-Universität Freiburg

Abstract: Die schnelle Inbetriebnahme von Rechnern, die flexible Nutzung von Maschinen mit verschiedenen Betriebssystemen und Softwareausstattungen sind selbst nach vielen Jahren noch nicht optimal gelöst. Hierfür bietet ein netzwerkbasierter Service mit einer breiten Auswahl zu startender Systeme und Installationsassistenten Abhilfe. Schnelle Datennetze, ein kompaktes Quickstart-OS auf Linux-Basis, ein zentraler Konfigurationsservice und geeignete Protokolle erlauben den einfachen Betrieb von Standardsystemen sowie die Erledigung von Wartungsaufgaben, wie Maschinen-Checks zur Fehlersuche, signifikant erleichtern. Es erfolgt eine Konzentration der Systemlogik auf einen zentralen Dienst, der je nach Anforderung mehrfach redundant ausgelegt werden kann.

Das Konzept basiert dabei auf dem bewährten Netzwerk-Booten, das für einen universelleren Einsatz verbessert wird. Hierzu werden einige Erweiterungen vorgeschlagen, die das Setup und die Konfiguration neuer und in Betrieb befindlicher Maschinen vereinfachen und in bestehende Infrastrukturen einbinden. Dabei wird die Beschränkung der engen Verknüpfung von PXE/DHCP/TFTP aufgehoben. Das erlaubt Rechner aus quasi beliebigen Netzen von einer Reihe verschiedener Bootmedien zu starten. Eine solche Minimalbootumgebung ließe sich für zukünftige Systeme vergleichbar zu den Fastboot-Installationen einiger Hardwarehersteller als Grundausstattung neuer Computergenerationen vorstellen. Auf der Netzwerkseite könnten der DFN oder einzelne Rechenzentren als Anbieter von neuen Diensten auftreten, die das Serviceangebot erweitern.

1 Motivation

Die Administration von Computern ist trotz aktueller Entwicklungen im Bereich des Software-Deployments immer noch aufwändig, erfordert eine Menge repetitiver Handlungen und bindet teures Personal. Vielfach dauert es sehr lange bis neu beschaffte Maschinen tatsächlich zum Einsatz kommen, was je nach Anzahl eine nicht unerhebliche Verschwendung von Ressourcen darstellt. Dabei unterscheiden sich die Problemstellungen zwischen verschiedenen Institutionen und Körperschaften nur unerheblich, so dass ein generischer Ansatz sinnvoll erscheint. Dieses Paper diskutiert für eine Reihe von Aufgaben ein Konzept, das sich die aktuellen und bewährten Trends der IT- und Netzwerk-Entwicklung zunutze macht. Aktuelle Trends zeigen in Richtung Zentralisierung: Home- und gemeinsame Verzeichnisse sowie die Authentifizierung werden dank schneller Netze überwiegend zen-

tral angeboten. Die konsequente Fortführung erlaubt eine Reihe von Automatisierungen:

- Schnelles Einbinden von neuen Maschinen in die bestehende Infrastruktur, Auswahl einer größeren Palette verschiedener (Linux-)Betriebssysteme für den Einsatz von Desktops, bis hin zu spezialisierten Cluster-Knoten im Grid-Betrieb
- Bereitstellung eines Kiosksystems oder Universaldesktops für Konferenzen, Tagungen oder Gastwissenschaftler
- Anbieten eines "Notsystems", wenn das eigentliche Betriebssystem von Nutzern nicht mehr korrekt arbeitet, was ein gewisses Weiterarbeiten erlaubt und eine Rettungsumgebung bereitstellt
- Erleichterung des Administration von großen Rechnernetzen durch das Anbieten von Spezialsystemen zur Datenrettung, Virussuche, Inventarisierung, der netzwerkgestützten Installation von Linux- und Windowssystemen oder der Bereitstellung einfacher Test- und Evaluationsumgebungen zur Evaluation von Hardware, dem Ausprobieren neuer Betriebssystem- und Softwarevarianten

Ausgehend von einem erprobten Remote-Boot sollen diese Dienstleistungen durch einen OS-on-Demand-Service bereitgestellt werden. Das vorgestellte System lässt sich sowohl lokal innerhalb einer Institution als auch als zentraler Dienst, der von mehreren Universitäten gemeinsam genutzt wird, realisieren. Das Modell ist auf verteilte Rollen angelegt, die es einzelnen Administratoren und Nutzern erlauben ausgehend vom Basis-System weitergehende Anpassungen vorzunehmen ohne dafür jedoch jede Maschine wieder komplett administrieren zu müssen.

2 Booten aus dem Netzwerk

Das Betriebssystem von der konkreten Maschine zu lösen, statt es auf der Festplatte zu installieren, erlaubt einen schnellen Austausch der Hardware nach Ausfällen, Bereitstellung einfacher Test- und Evaluationsumgebungen zur Bewertung der Maschinen-Hardware, dem Ausprobieren neuer Betriebssystem- und Softwarevarianten, den einfachen Umzug von Arbeitsplätzen oder die flexible Nutzung eines Computers mit verschiedenen Betriebssystemen. Rechner via Datennetz zu booten funktioniert seit geraumer Zeit und wird an der Universität Freiburg seit einigen Jahren für eine Vielzahl von Rechnern eingesetzt. Das ursprüngliche Remote-Boot wurde hierzu in den letzten Jahren am Rechenzentrum der Universität Freiburg weiterentwickelt und mit den Möglichkeiten der Virtualisierung speziell für Computer-Pools mit ihren vielfältigen Anforderungen optimiert[1]. Es entstand das Open Source Softwarepaket `OpenSLX`, das eine allgemeine Abstraktionsschicht für das Remote-Boot aus dem Netzwerk zur Verfügung stellt[2]. Dieses löst die bisher nur zu Teilen implementierten und sehr unterschiedlichen Ansätze der Linux-Distributionen für Netzwerkstarts oder Live-CDs ab. Gleichzeitig wird die Konfiguration der einzelnen Maschinen von der allgemeinen Bereitstellung eines gemeinsamen Root-Dateisystems für ein

[1] Siehe hierzu *http://www.ks.uni-freiburg.de/projekte/ldc*
[2] `OpenSLX`-Projektseite: *http://openslx.org*

bestimmtes Linux getrennt. Auf diese Weise arbeiten alle Client-Rechner ”stateless” und legen keine maschinenspezifischen Daten im lokalen Dateisystem ab.

Hierzu stellt `OpenSLX` eine Skriptsammlung bereit, die viele Standard-Distributionen – deren Auswahl sich mit überschaubarem Aufwand erweitern lässt – für den Stateless-Betrieb vorbereitet. Es bietet weiterhin eine Boot-Middleware, die für den Benutzer der laufenden Maschine optimalerweise unsichtbar bleibt. Das jeweilige typische Verhalten der gewählten Distribution bleibt erhalten, die Anpassungen und Änderungen finden im Hintergrund, hauptsächlich während des Bootvorgangs statt. Dieser wurde dabei so optimiert, dass er bestenfalls weniger Zeit als bei einer festplattenbasierten Installation benötigt.

`OpenSLX` definiert vier Stadien (Abbildung 1), welche die Schritte zur Einrichtung der Maschine sowohl auf dem Boot-Server als auch später auf den einzelnen Client-Rechner bezeichnen. STAGE1 beschreibt die primäre Installation einer Linux-Distribution durch Bootstrapen oder durch das Abbild einer Referenzinstallation auf einem weiteren Rechner oder einer virtuellen Maschine in ein Unterverzeichnis des Servers. Dieser Grundinstallation können lokale Anpassungen des Administrators oder Zusatzkomponenten hinzugefügt werden. Im anschließenden Schritt, dem zweiten Stadium, werden hieraus Root-Dateisystem-Exporte erzeugt. Dabei kann es sich um ein via NFS angebotenes Unterverzeichnis oder einen per Blockdevice bereitgestellten SquashFS-Container handeln. Beides erfolgt serverseitig zur primären Einrichtung und nach einem Update oder Änderungen des Root-Filesystems der Clients. Das Ganze ist so angelegt, dass sich auf einem Server theoretisch beliebig viele solcher Installationen anbieten lassen. Alle vom Administrator eingetragenen Exporte stehen sodann als Auswahlliste in einem `PXE`-Menü[3] bereit. Wobei sich selbstverständlich festlegen lässt, welches System nach einem definierten Timeout automatisch startet.

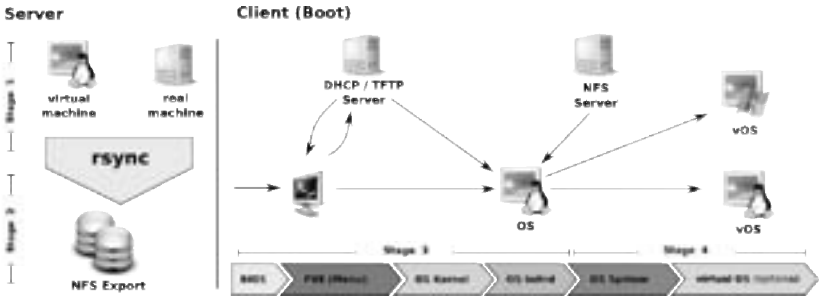

Abbildung 1: Stadien des klassischen OpenSLX

STAGE3 und 4 werden auf dem Client ausgeführt. Hierzu wird das von allen neueren Linux-Kernels unterstütze Konzept der *initramfs* genutzt. In diesem erfolgt das individuelle Setup des einzelnen Rechners durch von OpenSLX erzeugten Skripten. In STAGE3 läuft die individuelle Einstellung der Maschine ab und es werden die vom Administrator bestimmten lokalen Erweiterungen installiert. Im nächsten Stadium übergibt das OpenSLX-Init an das jeweils angepasste Runlevel-System der Distribution.

[3] `PXElinux` ist Teil des `Syslinux`-Pakets von H. P. Anvin: *http://syslinux.zytor.com*

2.1 Vom LAN-Boot ins Weitverkehrsnetz

Die meisten Remote-Boot-Lösungen spielen sich im LAN ab und setzen hierfür auf die weitverbreitete Pre-Boot-Extension (PXE). Diese setzt jedoch die Kontrolle über den lokalen DHCP-Server sowie einen TFTP-Server voraus. Das bedeutet oft eine erhebliche Einschränkung Maschinen in beliebigen Subnetzen aus dem Netz zu installieren oder zu starten. Die Einstellungen auf Subnetzebene mittels DHCP und TFTP-Konfiguration festzulegen, stellt eine nur unbefriedigende Lösung dar. Stattdessen wünscht man sich einen abstrakteren Ansatz, der Konfigurationen an zentralerer Stelle erlaubt. Die Entwicklungen im Bereich des Internets, des DFNs aber auch der Privatanschlüsse mit VDSL oder Kabel heben den alten Gegensatz von LAN und WAN zunehmend auf, so dass man zusätzlich zu den dargestellten klassischen LAN-Szenarien Rechner direkt aus dem Internet booten möchte.

Der Ansatz des PXE-Boots lässt sich jedoch nur bedingt auf das WAN transferieren. Oftmals hat der potentielle Benutzer gar kein Zugriff auf die notwendige Konfiguration des lokalen DHCP Servers – entweder aufgrund der Administrationsstruktur oder aber es fehlen die Einstellmöglichkeiten, wie beispielsweise bei den meisten Home-DSL-Routern. Um diese Problematik zu umgehen und von der Abhängigkeit über die Kontrolle von DHCP- und TFTP-Server zu lösen, läßt sich die Linux-Anwendung `kexec` verwenden. Sie erlaubt das Booten eines Betriebsystems oder bestimmter anderer Bootloader, wie `grub4dos` aus dem laufenden Betriebsystem. Das kann nun dazu verwendet werden die Aufgabe von PXE zu ersetzen. Es bedarf zu Beginn eines (minimalen) Linux-Systems mit Netzwerkunterstützung, dann können Kernel und die von OpenSLX generierte *initramfs* über das Internet heruntergeladen werden und mit `kexec` gestartet werden [SvS09].

2.2 WAN-Boot: Prototyp auf Basis des klassischen OpenSLX

Der eben vorgestellte Ansatz schafft eine Abhängigkeit: die eines laufenden Linux-Systems. Daher muss das zugrunde liegende Mini-Linux-System klein und flexibel sein. Es zeigt sich, dass sich ein solches System bestehend aus einem Kernel mit einer möglichst breiten Palette an unterstützten Netzwerkkarten sowie eines *initramfs* mit `kexec` in einem wenige Megabyte großen Image unterbringen lassen. In einem ersten Prototyp wurde das Erstellen der Bootmedien (CD-Image oder USB-Bootsektor) mit der Hilfe einer Web-Applikation, dem Pre-Boot-Server (PBS) erstellt [Sch10]. Dieser diente gleichzeitig als Gegenstelle für die auf dem Bootmedium untergebrachten Skripte. Zudem wurde OpenSLX so modifiziert, dass es bei der Erstellung der PXE-Menüs zusätzlich die Informationen über die bootbaren Systeme an die Web Applikation übergeben hat. Aus diesen Daten ließen sich dann in der Verwaltungsoberfläche des PBS Menüs zusammenstellen und auf einzelne Bootmedien abbilden. Zusätzlich konnte die Zuordnung der Menüs Abhängig von IP-Adressräumen gemacht werden. Für die Auslieferung des Kernels und der von OpenSLX erzeugten *initramfs* diente der PBS als HTTP-Schnittstelle zu dem lokalen TFTP-Server.

Mit dem PBS ließ sich ein erstes lauffähiges Model eines WAN-Boot Dienstes betreiben,

das jedoch Schwächen offenbarte. Das klassische OpenSLX entsprach nicht den Anforderungen eines flexiblen, vielseitig konfigurierbaren Dienstes.

3 Konzept: OS-on-Demand

Aus den Erfahrungen mit der auf OpenSLX basierenden Lehrpool-Umgebung an der Uni Freiburg und einem ersten WAN-Boot-Prototypen wird im Moment die nächste Generation des OpenSLX entwickelt, die erweiterten Anforderungen gerecht werden soll. Deshalb erfolgt eine logische Trennung von Maschinenkonfiguration und Bereitstellung der Root-Filesysteme für die verschiedenen Linux-Varianten. Die Konfiguration wird auf einem Server konzentriert und den lokalen Administratoren mit Filesystem-Overlays weitgehende Anpassungsmöglichkeiten eingeräumt.

Der bisherige Ansatz von OpenSLX sah vor, dass es in der Regel eine kleine Gruppe von Administratoren gibt, die selten die Konfiguration der Systeme ändern. Um jedoch einen über das lokale Netz hinausgehenden Dienst anbieten zu können, müssen die Komponenten zur Konfiguration der Systeme neu überdacht werden.

Bislang lag der Fokus der Entwicklung von OpenSLX in der Bereitstellung einer wartungsarmen, flexiblen Computerpool-Umgebung. Dies bleibt im neuen Konzept erhalten. Zudem sollte es möglich sein den Dienst einfach in bestehende Infastrukturen einzubinden. So sollen beispielsweise die Anbindung an die institutseigene Userauthentifizierung und die Einbindung der Heimatverzeichnisse der Benutzer funktionieren, ebenso wie die Erweiterung der Betriebssystemvorlagen durch zusätzliche Filesystemschichten (Overlays) mit vom Standard abweichenden Softwarepaketen. Dieses macht eine Mandantenfähigkeit der Verwaltungsoberfläche notwendig, damit die Administratoren der Institution in der Lage sind, Einstellungen und Erweiterungen für ihren Zuständigkeitsbereich eigenständig vornehmen zu können.

Konfigurierbarkeit für den Einzelanwender Der Einzelanwender soll die Möglichkeit haben, schnell und einfach "Live Systeme" auf seinem Rechner zu nutzen[4]. Zudem soll dem Benutzer gestattet werden eigene Einstellungen vorzunehmen, wie beispielsweise das Setzen eines Standardsystems beim Booten, das Einbinden von lokalen Speichermedien oder das Einrichten eines Autologins einer definierten X-Session. Für diesen Zweck soll es für Nutzer erlaubt sein, sich am Dienst zu registrieren, um die Einstellungen an zentraler Stelle zu hinterlegen.

Modifikationen der ursprünglichen Linux Distributionen Die notwendigen Veränderungen, die für die Vorbereitung einer üblichen Linux Installation zum Booten über das Netzwerk notwendig sind, haben sich seit dem Beginn der Entwicklung von OpenSLX massiv verändert. So muss ein Großteil der Hardwarekonfiguration nicht mehr manuell vorgenommen werden, da das System diese automatisch erkennt und einrichtet – so be-

[4] Vergleichbar mit dem Dienst *boot.kernel.org*

darf es beispielsweise im Normalfall keiner Konfiguration der X Windows Oberfläche mehr. Auf der anderen Seite gibt es neue Anforderungen, um ein System über das Netzwerk booten zu können. Die meisten der aktuellen Linux Distributionen optimieren den Startprozess ihrer Systeme massiv, beispielsweise durch die Verwendung von parallelisierten Init Systemen (z.B. *upstart*). Dies erschwert jedoch grundlegend die Integration des Netzwerk-Boots. Der Eingriff in das Init-System des zu bootenden Betriebssystems ist notwendig, da Aufgaben, wie das Netzwerksetup und das Einbinden des Root-Filesystems, die normalerweise das Betriebssystem eigene Init selbst ausführen würde, bereits durch die für den Netzwerk-Boot optimiertes *initramfs* ausgeführt wurden. Die meisten Modifikationen, die nötig sind, um ein System für den Netzwerk-Boot vorzubereiten, sind also nicht mehr Hardwareabhängig sondern Distributionsabhängig. Dies erlaubt die Modifikationen nicht wie bislang im von OpenSLX generierten *initramfs* auszuführen, sondern auf dem Server bei Bedarf zu erzeugen und für wiederholte Nutzung zwischenzuspeichern.

3.1 Komponenten

Aus den soeben beschriebenen Anforderungen ergibt sich der im Folgenden beschriebene Entwurf des OS-on-Demand(OSoD) Dienstes. Bevor auf diesen genauer eingegangen wird, werden die beteiligten Komponenten definiert:

PBL steht für Pre-Boot-Linux, es handelt sich hierbei um eine bereits in einer ähnlichen Form erprobte basierende Mini-Linux Umgebung. Sie hat die Aufgabe, die Funktion von PXE zu ersetzen und erweitern. Sie wird auf einem beliebigen Bootmedium ausgeliefert - das heißt sie kann auf USB-Sticks, CDs, Festplatten oder wiederum über PXE bereitgestellt werden. Das Bootmedium umfasst einen kompakten Linux Kernel, mit Unterstützung für die meisten Netzwerkkarten und der einer auf Busybox basierenden *initialramfs*. Neben den Standardkommandos zum Einrichten des Netzwerks und Nachladen weiterer Komponenten wird das PBL auch eine auf dem Linux-Framebuffer laufenden Client Applikation für die Interaktion mit dem Benutzer enthalten, welche direkt mit dem OSConfig-Server kommuniziert.

Der **OSConfig-Server** ist eine Web Applikation. Sie lässt sich in mehrere Komponenten unterteilen. Zum einen gibt es eine Komponente, die der Verwaltung des kompletten Systems dient. Sie beinhaltet das Rechtemanagement und verwaltet die zugehörigen OSRootfs Server/Proxies, die bootbaren Betriebssystem-Vorlagen als auch lokale Overlays – Erweiterungen zu den Basis-Betriebssystem-Vorlagen; zudem lassen sich hier neue Bootmedien erstellen. Eine weitere Komponente verwaltet die Einstellungen von Einzelbenutzern. Auf diese Komponente kann sowohl über ein Web Interface zugegriffen werden als auch über die im PBL enthaltene Client Applikation. Die letzte und entscheidende Komponente ist ein Interface zur Erzeugung und Verteilung von Kernel, Initrd und der Konfigurationslayer, die aus einer gewöhnlichen Linux-Installation ein über das Netz bootbares System machen. Dieses Konfigurationen werden im *initialramfs* des zu bootenden Systems abgerufen und über das nur lesbar eingebundene Root-Filesystem auf einen beschreibbaren

Layer (Unionfs[5]/Aufs[6]) gelegt.

Die **OSRootfs-Server bzw. -Proxies** dienen dazu, aus bestehenden Linux-Installationen Vorlagen-Images für den Netzwerk-Boot zu erzeugen. Diese werden mittels `rsync` kopiert, wobei gleichzeitig eine Vielzahl von laufzeitspezifischen Dateien bereits aussortiert werden, wie Log- oder Cache-Dateien. Diese "aufbereiteten Vorlagen" werden dann über das Netzwerk via NFS oder ein Network Block Device zur Verfügung gestellt. Die OSRootfs-Server erhalten ein Interface, um Vorlagen Images auszutauschen. Diese Schnittstelle werden auch die sogenannten OSRootfs Proxies, lokale Proxy Server für Vorlagensysteme, nutzen.

3.2 Aufbau

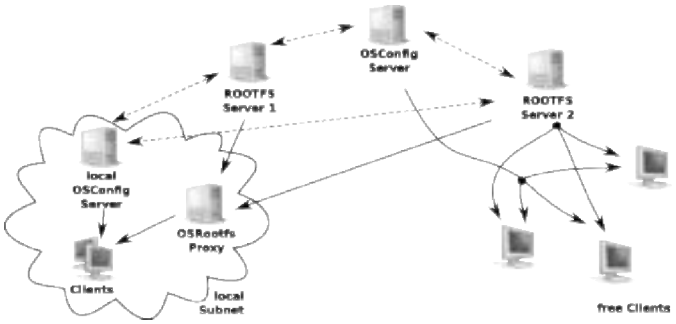

Abbildung 2: Aufbau des OS-on-Demand Dienstes

Das Zusammenwirken der einzelnen Komponenten veranschaulicht Abbildung 2. Prinzipiell lassen sich zwei Anwendergruppen unterscheiden: Institutionen (1) und der Einzelbenutzer (2). Institutionen haben in der Regel eine größere Anzahl von Rechnern zu verwalten und dabei komplexere Anforderungen an die Modifikationen der zu bootenden Linux Vorlagen, wie die Einbindung von Benutzerverzeichnissen und der zentralen Benutzerauthentifizierung. Zu diesem Zweck bietet sich der Betrieb eines eigenen OSConfig Servers an (3). Ein OSConfig Server ist mit einem oder mehreren OSRootfs Servern (5) verknüpft, da der OSConfig Server für die Erzeugung der Netzwerk-Boot Konfiguration Zugriff auf das per NFS vom OSRootfs Server eingebundene Root-Filesystem der zu bootenden Linux-Variante benötigt. Beim Betrieb einer großen Anzahl von Maschinen bietet sich auch die Verwendung eines OSRootfs Proxies (4) an, der alleine der Zwischenspeicherung der genutzten Vorlagensysteme im lokalen Netz dient. Die Einzelbenutzer des Dienstes können sich an einem öffentlichen OSConfig Server (6) registrieren und die von diesem vorgehaltenen Systeme aus dem WAN booten.

[5] Union File System *http://www.unionfs.org*
[6] Another Union File System *http://aufs.sf.net*

3.3 Bootprozess

Der Bootvorgang eines Clients unterscheidet sich an einigen Punkten vom klassischen Hochfahren eines lokalen Systems. Die Abbildung 3 zeigt schematisch sowohl den zeitlichen Ablauf als auch die dabei beteiligten Komponenten. Nach dem Einschalten des Rechners sucht zunächst das BIOS nach einem bootbaren Medium. Hierbei kann für das Booten einer OSoD Session eines der vom OSConfig erzeugten Bootmedien (1) verwendet werden. Es können – sofern vom Rechner unterstützt – optische Datenträger (CD/DVD), Flash-Speicher (SD/CF Karten), USB-Sticks, die lokale Festplatte oder auch PXE dienen. Von diesem Medium wird das Minimal-Linux PBL gestartet. Nach dem Netzwerksetup wählt der Benutzer mittels einer auf dem Linux-Framebuffer basierenden graphischen Benutzeroberfläche das zu ladende Betriebssystem aus, deren Inhalte die Applikation direkt vom OSConfig Server bezieht (2). Zusätzlich können an dieser Stelle Einstellungen geladen und verändert werden. Anschließend werden Kernel und *initramfs* des zu bootenden Systems geholt und mittels `kexec` gestartet. Das Programm `kexec` ist eine kleine Anwendung, die es erlaubt, ein Linuxsystem aus einem laufendenden System heraus zu booten, ohne dabei einen Neustart durchführen zu müssen. Es ist jedoch bis auf die Ausnhame der Kernel Command Line (KCL) nicht möglich (Konfigurations-)Daten in das neue System zu übermitteln. Die KCL ist eine längenbeschränkte Zeichenkette, mit der dem Kernel Parameter übergegeben werden können. In dieser platziert das PBL einen Sitzungsschlüssel, um zu einem späteren Zeitpunkt die passenden Konfigurationen vom OSCfonfig Server zu laden (3). Für die Erzeugung angepasster Konfigurationen besitzt der OSConfig Server Zugriff auf den OSRootfs Server (4). Nachdem der OS-Kernel geladen ist, wird die für das zu bootende System erzeugte WAN-Boot-Initramfs geladen. Diese ruft vom OSConfig Server den Konfigurationslayer ab und legt ihn über das nur lesbar vom OSRootfs Server eingebundene Root-Filesystem (5). Optional werden an dieser Stelle noch weitere Schichten in den Dateibaum eingebunden. Mit dem Verlassen des *initramfs* folgt der reguläre Systemstart. Um auch andere (proprietäre) Systeme über das soeben vorgestellte Verfahren laden zu können, ist es vorstellbar, nach dem Boot des zugrunde liegenden Linux Systems noch ein weiteres virtualisiertes Betriebssystem zu starten (6).

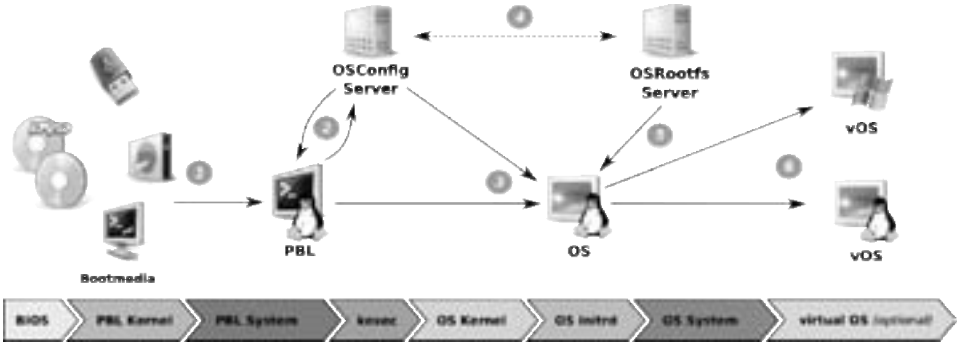

Abbildung 3: Bootprozess eines Clients des OSoD Dienstes

3.4 Verteilung der Root-Filesysteme

Bei der Umsetzung des vorgestellten Konzepts gilt es noch einige Fragen zu beantworten. Die zunächst wichtigste Aufgabe wird die Auswahl des geeigneten Wegs die Distributions-Vorlagen zu verbreiten. Im lokalen Netz hat sich hierfür NFS bewährt, jedoch zeigten die ersten Experimente mit dem Boot aus dem WAN, dass NFS hierbei an seine Grenzen stößt. Vielversprechender waren hierbei die Versuche mit Network Block Devices. Hierbei wird mittels Squashfs ein komprimiertes Loopdevice erzeugt, welches dann über ein entsprechendes Protokoll verteilt wird. Da diese Protokolle nur auf Blockebene arbeiten, besitzen sie wesentlich weniger Overhead. In Frage kommen hierfür das bereits im Kernel enthaltene NBD, sowie die am Lehrstuhl entwickelten Distributed Network Block Devices (DNBD, DNBD2) [Gre07].

4 Anwendungsfälle

Von den in der Motivation vorgestellten Einsatzmöglichkeiten sollen zwei vertieft werden, um die Verwendung des Konzept des OS-on-Demand zu verdeutlichen.

Universaldesktop für Veranstaltungen (Konferenz-Pools) Eine sich regelmäßig wiederholende Aufgabe ist die Bereitstellung von Desktopsystemen für Konferenz-, Tagungs- oder Seminarteilnehmer zum Online-Gehen. Dabei existieren nur wenige Unterschiede, an welcher Universität oder Forschungseinrichtung die Veranstaltung stattfindet. Hier bietet sich der Einsatz des WAN-Boots einer zentral vorgehaltenen Standardumgebung an, die bei Bedarf lokal erweitert wird. Ein wesentlicher Vorteil liegt darin, dass quasi beliebige Maschinen in einen solchen Konferenzpool abgestellt werden können, da keine Änderungen an eventuell existierenden Installationen auf der Festplatte vor oder nach der Veranstaltung vorgenommen werden. Dieses erübrigt auch das Vorhalten spezieller Maschinen, die nur zu diesem Zweck zum Einsatz kommen.

Dynamische Lehrpool-Steuerung Eine wirklich dynamische Steuerung der im Normalfall zu bootenden Betriebsysteme war mit dem klassischen LAN-Boot-Ansatz aufgrund der Vielzahl beteiligter Dienste, die je nach Institution in unterschiedliche Aufgabenbereiche (Administrationsdomänen) fallen, nicht trivial. Im Gegensatz zu den relativ statischen, über TFTP ausgetauschten PXE-Menüs bieten die dynamisch in der Web-Applikation *OS-Config Server* erzeugten Menüs eine Vielzahl von Varianten zur Manipulation. So lässt sich die Auswahl der bootbaren Systeme sowie die Defaulteinträge abhängig von Standort, Hardware, Bootmedium oder auch Uhrzeit steuern. In der denkbaren Kombination mit Techniken wie Wake-On-Lan besitzt der Administrator ein umfangreiches Werkzeug zur Steuerung seiner Pool Umgebung. Dies ermöglicht beispielsweise, dass zu einer bestimmten Uhrzeit ein vordefiniertes System für eine Lehrveranstaltung automatisch startet, dass Dozenten-PCs andere Voreinstellungen erhalten oder aber dass die Pool Rechner in ungenutzen Zeitintervalen andere Aufgaben erfüllen, wie über Nacht ihre Rechenkapazität in ein Cluster einzubinden.

5 Ausblick

Die präsentierte Lösung basiert auf langjährigen Erfahrungen im Bereich des Leih- und Lehr-Pool-, Cluster- und Desktopbetriebs am Rechenzentrum der Universität Freiburg. Während sich in statischen Umgebungen mit sehr fest umrissenen Aufstellungsorten und Maschinen die klassische PXE-Lösung anbietet, stößt sie für größere Installationen mit verteilten Managementrollen für die verschiedenen institutionellen Einheiten an ihre Grenzen. Sollten einzelne Nutzer zusätzlich die Möglichkeit besitzen, ihre Maschinen in einem gewissen Grade selbst zu verwalten, eignet sich ein zentraler Konfigurationsserver.

Der Linux-Diskless-Betrieb und die verteilte Administration des Root-Filesystems sind schon länger erfolgreich getestet. Für das WAN-Boot-Modul und den Konfigurationsserver hingegen existieren bisher nur Prototypen, die im Moment durch ein studentisches Teamprojekt verfeinert werden.

Das vorgeschlagene Betriebsmodell erhebt keinen Ausschließlichkeitsanspruch, sondern ist darauf angelegt mit anderen zu koexistieren. Kein Administrator muss die Hoheit über seinen DHCP-Server aufgeben oder von RIS Abschied nehmen. Ein Nutzer kann jederzeit sein lokal installiertes Betriebssystem booten und den OS-on-Demand Dienst lediglich zur Neuinstallation oder für Hardwaretests verwenden. Auf dem gezeigten Weg könnten die sich teilweise ausschließenden Softwareinstallations- und Managementinfrastrukturen unter einem einzigen Einstiegspunkt vereinigt werden.

Wenn man die Idee der bereits auf dem Mainboard oder Laptop vorinstallierten Minimal-Systeme[7] konsequent weiterverfolgt, dann könnte man sich in Zukunft ein universelles Mini-Boot-System (PBL) parallel dazu oder als Ersatz vorstellen. Die Platzanforderungen sind mit circa 15 MByte marginal. Ein Update ist nicht notwendig, da die relevante Hardware (Netzwerkkarte) fix ist. Das könnte Administratoren das Leben bei der Erledigung von Standardaufgaben erheblich vereinfachen und das Herumtragen und Aktualisieren von Installations- und Systemrettungsdatenträgern jeder Form überflüssig machen.

Literatur

[Gre07] Vito Di Leo Gremmelspacher. Development of a Distributed Network Block Device for Unicast Networks. *http://www.ks.uni-freiburg.de/download/bachelorarbeit/SS07/06-07-dnbd2-dileo/ba-dnbd2-dileo.pdf*, 2007.

[Gre08] Kate Greene. Booten ohne Verzögerung. *Technology Review*, 2008. *http://heise.de/-274854*.

[Sch10] Sebastian Schmelzer. Extending the playground. *http://www.ks.uni-freiburg.de/projekte/ldc/extendingtheplayground.pdf*, 2010.

[SvS09] Sebastian Schmelzer und Dirk von Suchodoletz. Network Linux Anywhere. *LinuxTag 2009*, 06 2009. *http://www.ks.uni-freiburg.de/projekte/ldc/networklinuxanywhere-paper.pdf*.

[7] Beispielsweise der Splashtop auf ASUS Geräten, [Gre08]