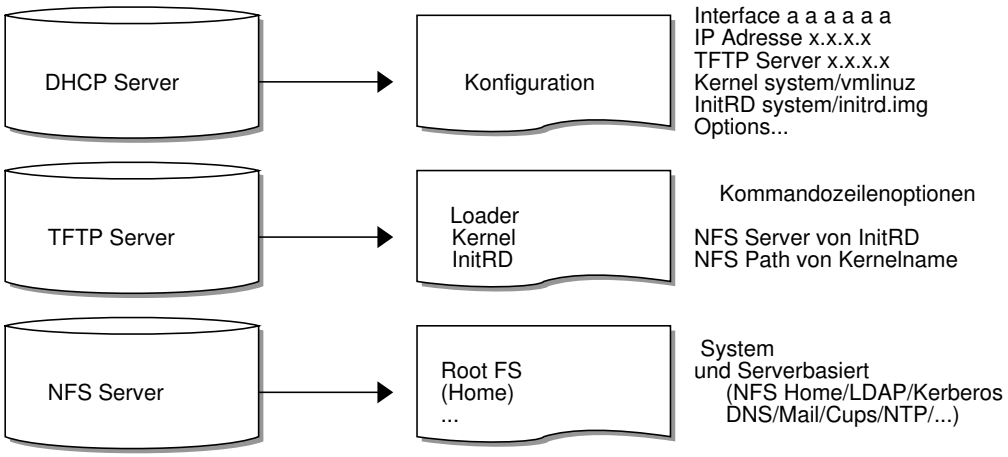
1 QSG Basic Netboot System

Für den Workstation-Cluster wird ein über das Netzwerk bootbares System benötigt. Dieses kann von einer normalen Festplatteninstallation auf den NFS-Server kopiert werden oder dort direkt erzeugt werden. Zweiteres wird hier beschrieben.



1.1 Vorbereitung

Wir benötigen einen DHCP, einen TFTP und einen NFS-Server. Dies kann durchaus eine Maschine sein. Auf dem NFS-Server werden die Pakete debootstrap und chroot benötigt.

1.2 NFS-Freigabe anlegen

Ein neues Verzeichnis wird für den NFS export angelegt und im Netzwerk freigegeben. Nur ein Host sollte hier Schreibrechte haben. So legen wir beispielsweise ein Verzeichnis für unser neues System an:

```
@nfs-server# mkdir -p /srv/remote/system-neu
@nfs-server# cat /etc/exports | grep /srv/remote/system-neu
/srv/remote/system-neu ↙
    ↳ 134.130.35.128/25(ro,no_root_squash,async,no_subtree_check) ↙
    ↳ 134.130.35.210/32(rw,no_root_squash,async,no_subtree_check)
@nfs-server# exportfs -rv
```

1.3 Basisinstallation

hier verwenden wir debootstrap um ein minimales System zu installieren. Der Kernel muss separat installiert werden.

```
@nfs-server# cd /srv/remote/system-neu
@nfs-server# debootstrap wheezy .
<... das dauert ...>
@nfs-server# echo "neu" > etc/hostname
@nfs-server# LC_ALL=C chroot . apt-get install linux-image-amd64
```

wenn das Netz innerhalb vom chroot nicht funtioniert, sind /etc/network/interfaces und /etc/resolve.conf zu inspizieren. Ein Blick in /etc/apt/sources.list kann auch nicht schaden, `contrib` und `non-free` stören hier nicht. Dateien können wir direkt auf dem Server editieren. Zum Installieren muss ins *chroot* gewechselt werde:

```
@nfs-server# echo "neu-chroot" > etc/debian_chroot
@nfs-server# LC_ALL=C chroot . bash
<... was zu tun ist und mit exit wieder raus ...>
@nfs-server# rm etc/debian_chroot
```

Wichtig nicht host und chroot verwechseln! Auch ist etc nicht /etc!!

Im *chroot* lässt sich die eventuell notwendige Firmware für einige Netzwerkskarten installieren.

```
@neu-chroot# apt-get update
@neu-chroot# apt-get upgrade
@neu-chroot# apt-get install pciutils less vim firmware-realtek firmware-bnx2
@neu-chroot# exit
```

Anmerkung Diese Vorgehensweise ist auch knackig wenn sich das Clientsystem mal nicht mehr aktualisieren lassen will, weil zb.Bsp. alle NFS Mounts vorher ausgehängt werden müssten (schon passiert).

Noch eine Anmerkung Nicht gleichzeitig chroot und das Clientsstem im rw-Modus starten. Es gibt ein Script (schroot) um dieses abzufangen.

Nun kopieren wir **das netboot** Script in das Konfigurationsverzeichnis für die initiale Ramdisk (*InitRD*) und diese konfigurieren. Wichtig sind die Einträge MODULES=netboot BOOT=netboot und das aufs-Modul! Dann können wir eine InitRD generieren:

```
@nfs-server# cp netboot etc/initramfs-tools/scripts/netboot
@nfs-server# grep -v ^# etc/initramfs-tools/initramfs.conf
<... hier fehlt der Output ...>     <-- TODO
@nfs-server# LC_ALL=C chroot . update-initramfs -k all -u
@nfs-server# ls -l vmlinuz initrd
lrwxrwxrwx 1 root root 30 Dec 20 19:31 initrd.img -> /boot/initrd.img-3.2.0-4-amd64
lrwxrwxrwx 1 root root 26 Dec 20 19:31 vmlinuz -> boot/vmlinuz-3.2.0-4-amd64
```

Seltsam initrd.img verlinkt hier auf das Hostsystem. Könnte man ändern, muss man aber nicht.

Die letzten beiden Dateien werden für für den TFT-Server benötigt.

1.4 TFTP-Server

Im TFTP-Rootverzeichnis wird ein Unterverzeichnis mit dem Namen des neuen Systems angelegt (hier. system-neu) in welches der Kernel und die zugehörige InitRD kopiert werden.

```
@tftp-server# mkdir <tftp-root-path>/system-neu
```

Kopieren, z.Bsp. mit scp:

```
@nfs-server# scp boot/vm* boot/in* sysop@tftp-server:<tftp-root-path>/system-neu
```

Wenn wir nicht die generischen Namen (cmlinuz und initrd.img) verwendet haben, verlinken wir diese auf dem TFTP-Server. Dies ist im Folgenden weniger Schreibarbeit.

```
@tftp-server# cd <tftp-root-path>/system-neu
@tftp-server# ln vmlinuz-* vmlinuz
@tftp-server# ln initrd.omg-* initrd.img
@tftp-server# ls -l
lrwxrwxrwx 1 root root 30 Dec 20 19:31 initrd.img -> initrd.img-3.2.0-4-amd64
lrwxrwxrwx 1 root root 26 Dec 20 19:31 vmlinuz -> vmlinuz-3.2.0-4-amd64
```

Nun fehlt hier noch der PXE-Lader `pxelinux.0` und eine zugehörige Konfigurationsdatei. Der Lader lässt sich hier http://ftp.nl.debian.org/debian/dists/squeeze/main/installer-amd64/current/images/netboot/netboot.tar.gz extrahieren. Diese beiden Dateien ghören im rootverzeichnis des TFTP-Servers:

```
@tftp-server# cd <tftp-root-path>
@tftp-server# cat pxelinux.neu
ipappend 2
label neu
    kernel system-neu/vmlinuz
    initrd system-neu/initrd.img
    append quiet
@tftp-server# ls -l pxelinux.*
-r-xr-xr-x 1 root root 30 Dec 20 19:31 pxelinux.0
-r-xr-xr-x 1 root root 30 Dec 20 19:31 pxelinux.neu
```

Wichtig ist der Eintrag `ipappend 1`, damit das Clientsystem bei PXE-Anfragen seine MAC-Adresse mit übermittelt bekommt. Bei verwendung einer älteren Versionenvon pxelinux.0 muss der initrd Eintrag in der Append Zeile mit übergeben werden (append initrd=system-neu/initrd.img quiet)

1.5 DHCP-Konfiguration

Nun muss nur noch der DHCP Eintrag auf dem DHCP-Server angepasst werden. Siehe dhcpd.conf:

```
@dns-server# cat /etc/dhcpd/dhcpd.conf
boot***;                 <-- noch nicht dokumentiert
authoritative;
ignore client-updates;
use-host-decl-names on;
option domain-name          "ti.rwth-aachen.de";      <-- Client Domain
option domain-name-servers  134.130.5.1, 134.130.4.1; <-- die DNS-Server
next-server                 134.130.35.193;           <-- der TFTP-Server
filename                    "pxelinux.0";
...
host hehe {                                           <-- Client Hostname
hardware ethernet 80:ee:73:40:6f:5x;                  <-- Client MAC
fixed-address 134.130.35.310;                         <-  Client IP
    option tiboot.configfile "pxelinux.neu";          <-- Client Bootoptionen/menu
    option root-path         "";       <-- optional
    }
```

Der next-server Eintrag meint den TFTP-Server. Bei modernen pxelinux Versionen heisst die configfile-Option: `pxelinux.configfile`[1]. Diese Datei kann zu einem kompletten Bootmenu ausgebaut werden um verschiedene Bootkonfigurationen starten zu können. Über `root-path` Könnte der in der InitRD vorgegebene NFS-Server oder das System angepasst werden. Die `nis-domain` Option wird zur Zeit nicht verwendet und könnte einfach dazu verwendet werden Parameter an das Initram-Script weiterzureichen, z.Bsp. zur initialisierung von verschiedenen Login Domänen für ssh.

Anmerkung Es sollte immer nur **ein** DHCP Server in einer Broadcast Domain aktiv sein. Mit ein wenig Trickserei (Aufteilung in verschiedene IP-Bereiche und/ oder Selektive Antworten auf DHCP und PXE Broadcasts) können auch zwei betrieben werden. Ist aber auch hochgradig vom Verhalten der Cliens abhändig und ein authoritative DHCP Server (also einer der auch mal nein sagt) ist so nicht mehr möglich. Die Fehlersuche dürfte auch sehr lustig werden.

Nun sollte ein Client booten können. Am besten wird erst einmal der rw-Client hochgefahren und das System eingerichtet (vim/locales/less/gnome/...)

1.6 Backup

```
    @nfs-server# mkdir system-bak
    (cd system-neu;tar c --one-file-system .) | ( cd system-bak;tar x )
```

1. Wir verweisen am besten auf die Dokumentation zu dhcpd

Snapshot auf dem NFS-Server (oder tarcopy verwenden).

1.7 Weitere Aufräumarbeiten:

Die Fehlermeldung *Driver 'pcspkr' is already registered, aborting...* wird durch das Laden des pc-speaker Treibers verursacht

```
echo 'blacklist snd-pcsp' >> etc/modprobe.d/blacklist.conf
```

Das Umbenennen von Netzwerksschnittstellen wird durch die udev Konfiguration verursacht. In

```
/etc/udev/rules.d/70-persistent-net.rules
#SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="*",
    ↳ ATTR{dev\_id}=="0x0", ATTR{type}=="1", KERNEL=="eth\*", NAME="eth0"
```

können alle Zeilen gelöscht/auskommentiert werden.

Einige Startupscripte werden nicht benötigt. So erfolgt der File-System Check auf dem NFS-Server und nicht auf den Clients. In

```
etc/init.d/checkfs.sh:# AJP not used
etc/init.d/checkroot-bootclean.sh:#AJP not used
etc/init.d/checkroot.sh:#AJP not used
etc/init.d/hostname.sh:# AJP not used
```

kann am Anfang mit `exit` das Skript beendet werden.

1.8 Literaturhinweise

- PXELinux Debian
- PXE-Boot mit NFS
- Debian Diskless
- NFSv4 Arch Linux
- Diskless Ubuntu RW
- Initramfs scripts
- PXELinux.0

———1———2———3———4———5———6———7———8———9———

```
    SYSTEM=${PWD # #*/}
    #scp a,b  TFTP-Server:/TFTP-path/$SYSTEM/
    scp $KERNEL  sysop@oxy:/remote/template/tftpboot/$SYSTEM/
    scp $INITRD  sysop@oxy:/remote/template/tftpboot/$SYSTEM/
```

```
#new irfs
sed -i 's,^MODULES=most,MODULES=netboot,' etc/initramfs-tools/initramfs.conf
grep MODULES etc/initramfs-tools/initramfs.conf


schroot . passwd
schroot . adduser --home /var/sysop --uid 999 -gid 4 sysop
  LC_ALL=C apt-get install console-setup keyboard-configuration #choose de-nodead
  LC_ALL=C dpkg-reconfigure keyboard-configuration
      LC_ALL=C apt-get install locales #choode {de_DE,en_GB}.UTF-8
  LC_ALL=C dpkg-reconfigure locales
  apt-get install vim gpm

apt-get install pciutils #lspci
 #apt-get inatall console-setup-mini # keyboard, mayb w/o mini    FUCK - dnw!!!!!
 #udevadm trigger --subsystem-match=input --action=change
 #apt-get install xkb-data
```

```
cp netboot etc/initramfs-tools/scripts/netboot
echo aufs >> etc/initramfs-tools/modules
sed -i -e 's,^BOOT=,BOOT=netboot,' 's,^MODULES=,MODULES=netboot,' ↙
    ↳ etc/initramfs-tools/initramfs.conf
echo NFSROOT=134.130.35.204:/srv/remote/* >> etc/initramfs-tools/initramfs.conf
chroot . update-initramfs -k all -u
```