Diskless Booting via AFS

Summary of Advantages of AFS over NFS
as a network subsystem for diskless booting
and
a Micro-Mini-HowTo

Motivation

- Diskless booting in general
  - Ease of management
  - Cost (disk hardware, OS upgrades)
- NFS vs AFS
  - Reliability
  - Reduce disk space requirements (50MB vs 1MB per server)

Goals of Diskless Booting

- Ease of Configuration
  - stock kernel/modules
  - Standardization (PXE or nbi)
- Clean and sensible
  - “Normal” appearance to end user
  - Not too many weird error messages while booting
- Reliable
  - Use AFS redundancy
  - Use AFS caching

Issues with NFS network boot

- Client is dependent on one boot server to be up and running all the time
- NFS over UDP can be flaky under heavy loads
- 50MB–100MB of disk storage per client
- Log files can generate a lot of chatty network traffic and disk I/O on the boot server.

Roadmap to Diskless Booting

- Kernel with nfs built-in, saved on a floppy, NFS
- Kernel with nfs built-in, PXE/etherboot, NFS
- Stock kernel, hacked initrd, PXE/etherboot, NFS
- Stock kernel, hacked initrd, PXE/etherboot, AFS
- Stock kernel, hacked initrd with lots of network modules, PXE/etherboot, AFS – One size fits all

Tools to Make AFSBoot Work

- New mount --bind (2.4 kernel)
- New tmpfs filesystem (2.4 kernel)
- Better dhcp client – dhcpcd
- Working PXE for Linux
  PXE support in DHCP server

Issues with AFS network boot

1. Getting AFS running before we have a root file system (chicken/egg problem)
2. Configuring AFS parameters (cache size, boot server) on the fly
3. Device files
4. Clean shutdown–keeping AFS running until the very end
5. Security (suid files, protecting ssh keys, etc.)

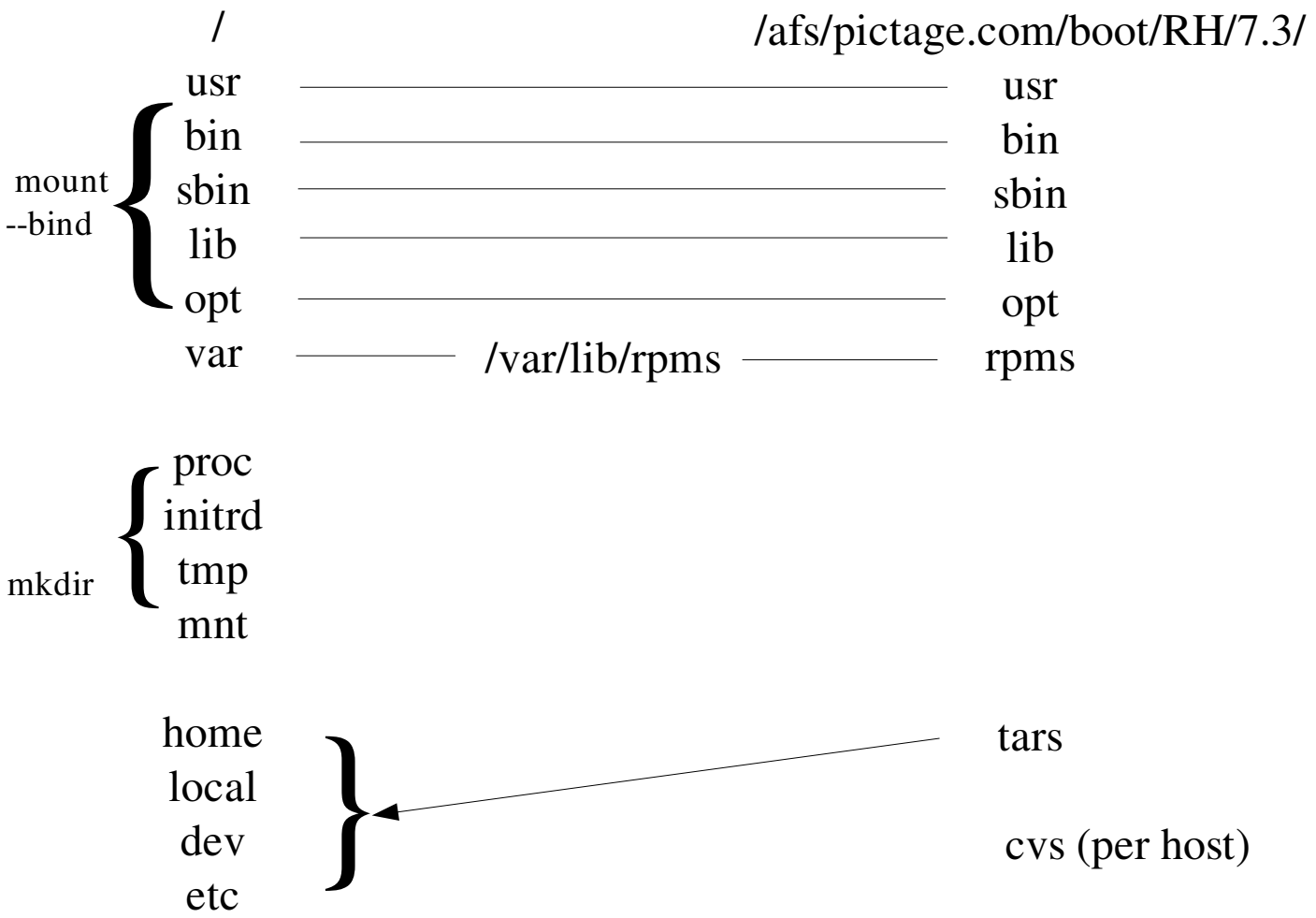
Solutions to AFS network boot

1. Use initrd to get libafs*.o module loaded
2. Get parameters from dhcpcd. Parse rootpath.
3. Put non-readonly files (root filesystem) into a tmpfs filesystem. (etc, dev, var, tmp)
4. Hack system scripts (ifdown, netfs) to keep network interface running until poweroff
5. Suid files unnecessary. Use auto_klog to get appropriate tokens.

Starting AFS in initrd

- Load appropriate network module
- Initialize NIC with dhcpcd
- Create /sysroot as tmpfs
- Copy AFS files to /sysroot
- Load kernel module
- /sysroot/usr/vice/etc/afsd -memcache
- mount --bind top level read-only directories
- Create remainder of root filesystem from tarballs

Building a Filesystem



Magic linuxrc file

```
#!/bin/bash


echo Preloading one network module
/sbin/modprobe -t net \* 2>/dev/null

echo Mounting /proc filesystem
mount -n -t proc /proc /proc

echo Starting loopback device
ifconfig lo 127.0.0.1 up

echo Configuring network interface
dhcpcd eth0
killall -KILL dhcpcd
. /etc/dhcpc/dhcpcd-eth0.info

<get params from $ROOTPATH>
```

```
echo "Mounting / filesystem as tmpfs"
mount -n -t tmpf
   -o
size=90M,mode=755,uid=0,gid=0,nr_inodes=18k
   tmpfs /sysroot

echo "Building directories and mountpoints"
for dir in afs bin boot dev etc initrd lib mnt opt
            proc ram sbin usr tmp var/lib/rpm; do
   mkdir -p /sysroot/$dir
done

echo "Starting AFS system"
cp -a /usr/vice /sysroot/usr
echo $CELL > /usr/vice/etc/ThisCell
# Load AFS module for this kernel
insmod /sysroot/usr/vice/etc/modload/libafs-2.4.20-
20.7.mp.o
/sysroot/usr/vice/etc/afsd -blocks $CACHE_SIZE
-memcache -nosettime -mountdir /sysroot/afs

echo "Mounting AFS readonly mountpoints"
for dir in bin lib opt sbin usr var/lib/rpm; do
   mount --bind /sysroot$ROOTPATH/$dir /
sysroot/$dir
done

/sysroot/bin/chmod 1777 /sysroot/tmp
```

Odds and Ends

/etc/init.d/halt

```
halt_get_remaining() {
  awk '!/(^#|afs|proc|loopfs|autofs|devfs|^none|
^\/dev\/root| \/[^/]* nfs ro,| \/ )/ {print $2}' /proc/mounts

  awk '{ if ($3 ~ /^proc$/ && $2 !~ /^\/proc/) print $2; }
' /proc/mounts

}
```

/etc/init.d/network

```
# shut down all interfaces (other than loopback)
for i in $interfaces ; do
    eval $(fgrep "DEVICE=" ifcfg-$i)
    if [ -z "$DEVICE" ] ; then DEVICE="$i"; fi

   if LANG=C egrep -L "^ONSHUT=\"?[Nn][Oo]\"?"
ifcfg-$i > /dev/null ; then
      # Don't shut down. Keeps AFS root running.
      # Do zap the dhcpcd daemon for this interface
      if [ -f /etc/dhcpc/dhcpcd-eth0.pid ]; then
         echo "Killing dhcpcd daemon ONLY for $i"
         kill -KILL `cat /etc/dhcpc/dhcpcd-eth0.pid`
      fi
      continue
   fi

   if ! check_device_down $i; then
       action $"Shutting down interface $i: " ./ifdown $i
boot
   fi
done
```