Introduction to Network Booting

Free Yourself From Your Hard Drive!

Joshua Oreman

Student Information Processing Board
Massachusetts Institute of Technology

Independent Activities Period, January 6, 2010

Outline

Q&A throughout the presentation

A typical large site

- Many, many similarly- or identically-configured systems
- Each boots up independently, from its own hard drive, and then tries to access shared network resources
- Very fast, powerful network
- Nothing gets done if the network is down
- Most interesting user/business data is stored on the network, not locally
- High-performance, finely tuned central servers already essential to getting things done (Kerberos, AFS, mail, ...)

The problem with local booting

A supercomputer cluster...

- Local drives used pretty infrequently
- Local drives fail very often
- And they're expensive!

An office building...

- Employees like to customize their systems
- Distributed updating is hard

MIT!

- New systems constantly needing identical configuration, installation
- Users' machines incompatible with course needs

The big idea

Get rid of the client machines' hard drives entirely!

How it works

- Instead of starting from the hard drive, clients execute a small bootstrap program in ROM
- Bootstrap program autoconfigures the network, fetches the operating system kernel, and executes it
- OS kernel mounts a networked drive as its main system disk

Advantages for large sites

- Files that are the same for everyone only exist in one place: upgrading is a breeze
- One more fancy central server is more than paid for by saving lots of consumer-level hard drives

The big idea

Get rid of the client machines' hard drives entirely!

Advantages for occasional use

- Clients can netboot into a special environment without making any inconvenient or dangerous changes to their hard drive
- Diagnostic tools and system installers can be run without carrying around a bunch of CDs

But most of all...

it's fun!

Networking basics

A computer network is layered:

- Link layer: transports data frames between physically connected hosts; Ethernet
- Internet layer: handles routing packets to hosts on other networks; Internet Protocol (IP)
- Transport layer: provides data integrity, multiplexing, and connection semantics;
  - Complex: Transport Control Protocol (TCP) creates a reliable data stream between hosts. Everything arrives, in order, without duplication.
  - Simple: User Datagram Protocol (UDP) provides unreliable transmission of independent packets. May be lost, reordered, or duplicated.
- Application layer: handles the useful part of the connection, file data or email or ...

Network configuration for local boot

- Dynamic Host Configuration Protocol
- Client knows nothing about its role in the network
- Server supplies network configuration to each computer
- Differentiate between systems by MAC address

```
Internet Systems Consortium DHCP Client V3.0.6
Copyright 2004-2007 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:16:3e:58:d1:cf
Sending on   LPF/eth0/00:16:3e:58:d1:cf
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 3
DHCPOFFER of 18.181.1.231 from 18.181.0.221
DHCPREQUEST of 18.181.1.231 on eth0 to 255.255.255.255 port 67
bound to 18.181.1.231 -- renewal in 41858 seconds.
```

Figure: A DHCP session

Changes for network booting

- The same protocol (DHCP) is used to get IP, DNS, router
- Additional DHCP options fill in the gaps:
  - Server to load files from (`next-server`)
  - Filename of kernel to load (`filename`)
  - OR Remote disk to boot from (`root-path`)
- With this information the bootloader can access the network normally
- TFTP: very *simple* protocol to fetch image
- Proxy DHCP: extension allowing different servers to specify network configuration and booting information

Network attached storage (SAN)

Server provides a hard disk image to clients.

- Very light CPU load on the server, but clients gain benefit of server's cache
- High-speed RAID and a fast network can give better performance than consumer hardware
- Windows can't boot from a network filesystem, but can boot from SAN
- But two clients can't write to the same image

Often a hybrid approach is best: SAN for the root filesystem, NFS/AFS for user data

Copy-on-write

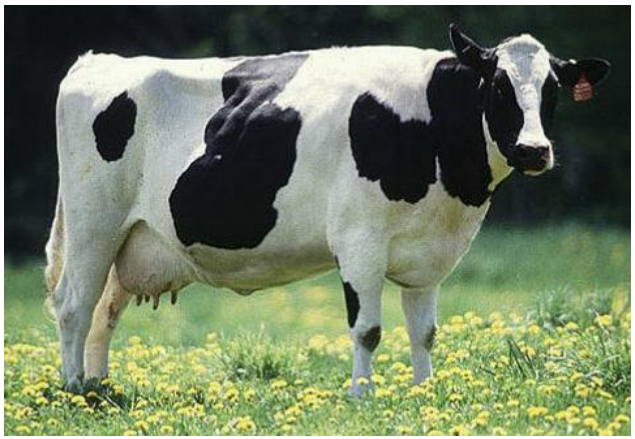


Figure:
If at all possible, involve COW.

- Copy-on-write (COW) makes SAN booting easier
- One read-only image is provided to all clients, but it looks writable.
- Writes actually go to a separate file, unique to each client, that only contains the modified information.
- Reads come from the COW file if the block is there, the base image if not.
- Upgrades: change the base image and delete all the COW files.

The BIOS and option ROMs

- BIOS is the very first thing run when you power on, burned into a chip inside your computer
- Main job is to initialize hardware, catalogue boot devices and start an operating system from one
- Since the BIOS doesn't know everything, option ROMs allow its functionality to be extended
- ROMs called during power-on self test to extend or replace BIOS functionality
- Hook interrupts:
  - Provide access to new types of hard disk (`int 13h`)
  - Trap the "boot system" call to netboot (`int 19h`)
  - On newer systems, ROMs can register as boot devices and appear in menus

Initial system bootstrapping

Initial loading occurs in real mode, so it can use the BIOS and option ROMs.

- Goal: load enough for the OS to manage the rest on its own
  - Kernel image
  - Device drivers for hard disk or network card
  - Filesystem drivers
  - Network layer
- For Linux, drivers are in an initial ramdisk (`initrd`) containing code to load them and mount a root filesystem
- For Windows, there's a way of marking drivers as boot-loaded

Pivoting to a full system

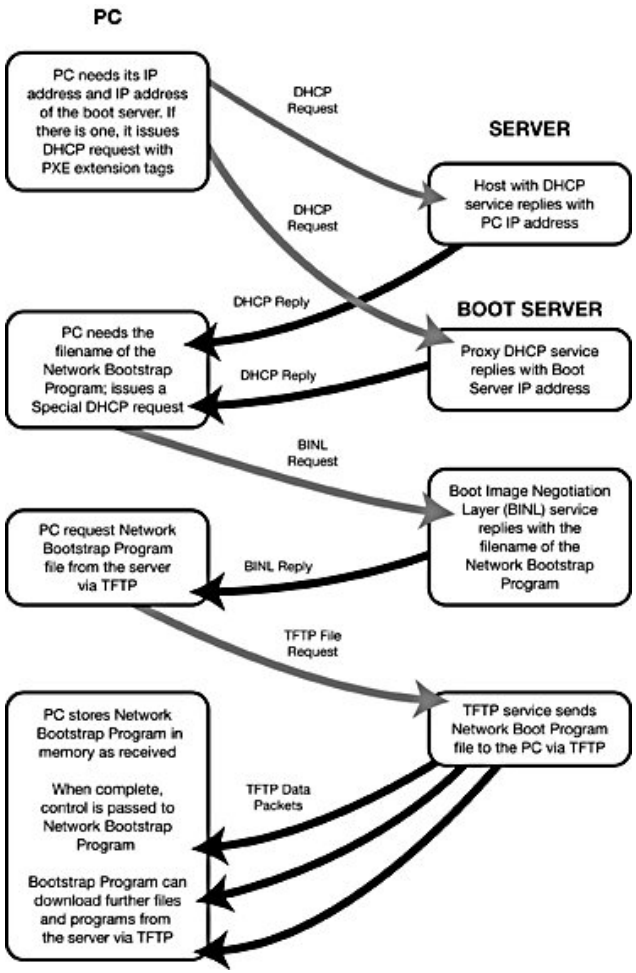
To do anything useful, the OS must switch to protected mode, where it can't use the BIOS or ROM services.

- Once real-mode loading is done, the loaded drivers are used to let the OS access its boot device for itself.
- Results in much better performance—boot-time drivers are as simple as possible because of size limitations
- Avoids resource conflicts—OS kernel can manage everything
- But how does the OS know where to look?
  - Kernel command line: user specifies all info in bootloader configuration
  - Boot firmware tables: bootloader leaves info in a special table the kernel can search low memory for

Ad-hoc simplicity: Net Boot Image Proposal (1993)

- Very first attempt to describe a mechanism for netbooting
- Bootstrap loader fetches network boot image (NBI) via TFTP
- NBI describes how to transfer control to kernel and bundled drivers
- No callback interface
- 1995: Etherboot released, capable of booting NBI images

Corporate complexity: Intel's Preboot eXecution Environment (1997)



- Instead of a filename, DHCP options contain a boot menu
- Chosen option yields server to request filename from
- The named Network Boot Program is downloaded using TFTP
- Callback interface allows the NBP to access the network and download more
- *Horribly* specified, but it's Intel
- PXE is standard now

PXE's problems

- Poor specification leads to buggy implementations: nobody does things in quite the same way
- Limited to TFTP, which is slow and does not scale well
  - Lower-level options: UDP or Ethernet, no IP!
  - Any NBP wanting to load files from a web server would have to implement a full TCP/IP stack
- NBPs officially limited to 32kB
- Net Boot Image format was lightweight, easy to automatically "wrap" a kernel and drivers into one bootable file;
  NBP must be a program that locates and downloads those components!

An influx of sanity: PXELINUX (1999)

```
# tftp://installer.mit.edu/debathena/pxelinux.cfg/default

default Debathena
label Debathena
  kernel vmlinuz
  append netcfg/get_hostname= locale=en_US console-setup/layoutcode=us interface=auto \
url=http://18.9.60.73/installer/jaunty/debathena-jaunty.preseed initrd=initrd.img
```

Figure: The `pxelinux.cfg` file for MIT's networked Debathena installer

- Part of the SYSLINUX suite; unified configuration and module format, complicated/polished menuing possible
- Very easy to boot a Linux kernel, load an initrd, and specify the kernel command line
- And there was much rejoicing.

Ten years later...

- PXE has never been updated, largely becaue Microsoft doesn't like centralized booting
- Networks are much faster (gigabit Ethernet) and more ubiquitous
- "Thin client" applications are more popular
- People just want to do more with netbooting

to the rescue!

gPXE to the rescue!

gPXE: an extensible, powerful, open-source network bootloader containing:

- Drivers for dozens of network cards, including almost all those in common use
- Stacks for Infiniband and 802.11 wireless
- A chainloader for vendor PXE that uses its network driver
- A complete, robust PXE stack and TCP/IP stack
- A DNS resolver so you can use hostnames
- Download protocols: TFTP, HTTP, HTTPS, FTP, NFS
- SAN boot support: iSCSI, AoE, Infiniband SRP
- Image formats: ELF, COM32, Multiboot, NBI, PXE, Linux
- A command line with scripting support

Booting from a SAN disk

```
# dhcpd.conf
option root-path "iscsi:rwcr.mit.edu:::iqn.2009-05.net.rwcr.xenon:dosboot";

gPXE> sanboot aoe:e1.3
```

Figure: Two ways of configuring a SAN boot

- Invoked from DHCP `root-path` option or gPXE `sanboot` command
- SAN boot protocols:
  - iSCSI: routable, reliable, complicated, some overhead
  - AoE (ATA over Ethernet): unroutable, less reliable, very simple, fast blast
- Naming a disk to boot from: the root path
  - iSCSI: server:protocol:port:lun:iqn
  - Initiator Qualified Name: date and DNS establish ownership
  - AoE: shelf.slot

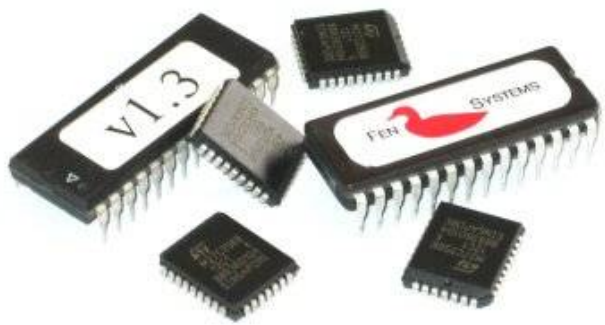
Booting with a gPXE script

```
PXELINUX using HTTP
#!gpxe
set 209:string http://my.server/netboot
chain ${209:string}/pxelinux.0

gpxelinux
#!gpxe
dhcp net0
imgload pxelinux.0
boot

Wireless boot at MIT
#!gpxe
set net0/ssid MIT
autoboot

Complex boot with static IP
#!gpxe
set net0/ip 18.181.1.231
set net0/gateway 18.181.0.1
set net0/netmask 255.255.0.0
set net0/dns 8.8.8.8
ifopen net0
kernel http://my.server/vmlinuz
initrd http://my.server/initrd.php?nic=${net0/mac}
boot
```

PXE chainloading

- Almost every network card in existence has a vendor PXE ROM
- Idea: gPXE packaged as a PXE Net Boot Program, so gPXE's features are available without burning it into ROM
- UNDI driver allows a special gPXE (`undionly.kpxe`) to work even on network cards it doesn't have a driver for
- Configure DHCP server to hand out `undionly.kpxe` to clients with no user class, and the real boot file (e.g. a gPXE script) to clients with user class `gPXE`

Option ROMs



- Stored on a flash chip in an expansion card, or in the BIOS
- New images can be flashed
- ROM identifies the PCI device it's associated with

- BIOS and ROMs execute in real mode, so only 1MB of memory is accessible.
- Total size of ALL ROMs cannot exceed 128kB! Extremely limiting.
- Partial solution: allocate more space with POST Memory Manager (PMM), and move code there to make room
- New systems with PCI 3.0 can do even better

Demonstrations

- Rescue disk boot over the Internet
- Debathena networked installer using HTTP
- Linux iSCSI boot
- DOS MEMDISK boot with network card ROM flashing
- Authenticated boot menus

- http://boot.kernel.org/
- http://netboot.me/
- Encrypted wireless boot
- Windows AoE boot with system BIOS splicing
- Wireless boot from BIOS
- Networked Debathena liveCD

What lies ahead

- In the works:
  - Better scripting
  - SAN ISO booting support
  - Move some functionality into modules
  - Better documentation
- A glimmer in the developers' eyes:
  - Load files out of filesystems on SAN disks
  - Boot your XVM on a local machine
  - "Netbooting for everyone"
  - Josh's Crazy Virtualization Idea

Wrap-up

Much more information is available at http://etherboot.org/

Think this stuff is awesome? Get involved!
Stop by the SIPB office sometime, W20–557.

Want to get paid for working on gPXE?
Look for the Etherboot Project in Google Summer of Code 2010.

Thanks to Marty Connor for providing demo hardware, assisting with demos, and being an all-around great project leader!