PXE network boot methods

Luc Sarzyniec
<devel@olbat.net>

2013-02-21

Contents

1 Introduction to PXE boot

1.1 Boot over the network procedure

1.2 The PXE specifications

1.3 Network Bootstrap Programs

1.4 Configure a specific NBP on DHCP server

1.4.1 ISC DHCP server

1.4.2 dhcpcd

1.4.3 dnsmasq

2 Network Bootstrap Programs

2.1 PXElinux

Features

- Download files from TFTP
- Boots downloaded Linux kernels images
- Boots COMBOOT images
- Comes with different COMBOOTs (chain.c32 allow to chainload on hard disk's partitions)
- Customizations possible with DHCP options
- Boots nodes following a node-specific profile
- Chainload other NBPs

Pros and Cons

⊕ Standard

⊕ Stable

⊕ A lot of custom COMBOOTs

⊖ Do not work with HTTP

⊖ Cannot boot local systems (the only solution is to chainload an installed bootloader with a COMBOOT)

⊖ Hard to embed on a NIC's PROM, so it's necessary to download the NBP at every boot

2.2 GPXElinux

Features Same as PXElinux, but it adds the support of HTTP and FTP.

2.3 iPXE

Features

- Download files from TFTP/HTTP/FTP
- Boots downloaded Linux kernels images
- Interactive shell
- Manual configuration of NIC/Network
- Scripting
- Boot COMBOOT images (specific build)
- Embedable on NIC's PROM
- SAN compatible
- Secure authentication settings
- Logs on syslog server
- Can boot nodes following a node-specific profile
- Chainload other NBPs

Pros and Cons

⊕ Embedable on NIC PROM

⊕ Interactive shell

⊕ Scripting

⊕ SAN compatible

⊕ Secure authentication settings

⊕ Logs on syslog server

⊖ Support of COMBOOTs is disabled by default

⊖ Cannot boot local systems (the only solution is to chainload an installed bootloader with a COMBOOT)

2.4 GRUB2 disk

Features

- Download files from TFTP/HTTP
- Boot downloaded Linux kernels images
- Boot of local systems (support of a lot of OS brands)
- Interactive shell
- Manual configuration of NIC/Network
- Scripting
- Can boot nodes following a node-specific profile
- Chainload other NBPs (buggy)

Pros and Cons

⊕ Boot of local systems (support of a lot of OS brands)

⊕ Interactive shell

⊕ Scripting

⊖ Not a main feature of the software (not really stable)

⊖ Not embedable on a NIC's PROM, so it's necessary to download the NBP at every boot

2.5 Comparison of NBPs

	PXElinux	GPXElinux	iPXE	GRUB2 disk
Download via TFTP	×	×	×	×
Download via HTTP		×	×	×
Download via FTP		×	×	
Boot downloaded kernel images	×	×	×	×
Load COMBOOTs	×	×	×	
Embedable on NIC's PROM	∼	∼	×	
Boots nodes following a profile	×	×	∼	∼
Chainload other NBPs	×	×	×	∼
Interactive shell			×	×
Scripting			×	×
Chainload another bootloader	×	×	×	×
Boot of local systems				×
SAN compatibility			×	
Secure authentication settings			×	
Logs on syslog server			×	

3 Install and configure NBPs

3.1 PXElinux/GPXElinux

The software is packaged on most of Linux distros, the NBP files are included in the package *syslinux*, you can also download the software at `http://www.kernel.org/pub/linux/utils/boot/syslinux/`.

After you got the sources of PXElinux, you'll just have to copy the NBP file *pxelinux.0* (or *gpxelinux.0*) inside your PXE repository (TFTP or HTTP server). It's also necessary to configure your DHCP server to make it load this NBP (see section 1.4).

To understand how this NBP will boot the nodes depending on a specific profile, take a look to the section 4.

3.2 iPXE

The git repository of the software is available at `http://git.ipxe.org/ipxe.git`, you can have more information on the download of iPXE at `http://ipxe.org/download`.

The source code (that is used to compile specific iPXE files) is available under the directory `src/`.

3.2.1 Boot a iPXE NBP from DHCP

You can generate an iPXE NBP named `undionly.kpxe` from the sources if you don't want to flash your NIC's. This file has to be loaded at boot

time by the PXE-compatible NIC of nodes so it's necessary to configure the DHCP server to make this file be loaded at boot time (see section 1.4).

To generate `undionly.kpxe`, compile the file with:

```
make bin/undionly.kpxe
```

If you want to embed a configuration script in your NBP (what will be pretty useful in the section 4), you can compile *undionly.kpxe* with the command:

```
make bin/undionly.kpxe EMBED=bootscript.ipxe
```

Where `bootscript.ipxe` is a file containing the iPXE script.

Further information are available on iPXE's website at `http://ipxe.org/howto/chainloading` and `http://ipxe.org/embed`.

3.2.2 Create an iPXE ROM to be burned on a NIC's PROM

To create a iPXE ROM that'll be burnable on a NIC's PROM, it's necessary to know the device ID and the vendor ID of the NIC.

Under Linux, you can get the device ID in the `/sys/class/net/IFACENAME/device/device` file and the vendor ID in the `/sys/class/net/IFACENAME/device/vendor` file (`IFACENAME` is the name of your NIC, remove the prefix 0x in the following commands).

You can also use *lspci* as specified in `http://ipxe.org/howto/romburning`.

After you got this two IDs, you can compile the rom using the command:

```
make bin/VENDOR_IDDEVICE_ID.rom
```

The vendor and device's IDs have to be concatenated and the hexadecimal number should be in lower case.

If you want to embed a configuration script in your NBP (what will be pretty useful in the section 4), you can compile the rom with the command:

```
make bin/VENDOR_IDDEVICE_ID.rom EMBED=bootscript.ipxe
```

Where `bootscript.ipxe` is a file containing the iPXE script.

To burn this ROM on your NIC's PROM you can follow the instructions on iPXE website: `http://ipxe.org/howto/romburning/intel`.

Further information are available on iPXE's website at `http://ipxe.org/howto/romburning` and `http://ipxe.org/embed`.

3.2.3 Load iPXE from GNU/GRUB

It's possible to generate a iPXE kernel file that can be loaded from GNU/GRUB. This file can be stored on the local hard-disk of the node.

First of all, it's necessary to compile this kernel file (`ipxe.lkrn`) with the command:

```
make bin/ipxe.lkrn
```

Then you have to copy this file somewhere of the node's hard disk (let's say inside `/boot/ipxe.lkrn` witch is based on the partition number `#P` of the disk number `#D`.

The last step is to write update the GNU/GRUB configuration file to make it load this kernel.

Under GRUB legacy

The configuration file should look like:

```
timeout 10
default 0
title iPXE
  kernel (hd#D,#P)/boot/ipxe.lkrn
```

Where `#D` and `#P` have to be replaced by the disk and partition number, for sample: `(hd0,1)`.

If you want to add a custom iPXE script, just save a *plain-text* file containing the script somewhere on the disk (let's say in the `/boot/bootscript.ipxe` file), then add the line `initrd /boot/bootscript.ipxe` after the `kernel` entry in the configuration file:

```
timeout 10
default 0
title iPXE
  kernel (hd#D,#P)/boot/ipxe.lkrn
  initrd /boot/bootscript.ipxe
```

Under GRUB 2

The configuration file should look like:

```
menuentry iPXE {
  set root=(hd#D,#P)
  linux16 /boot/ipxe.lkrn
}
```

Where `#D` and `#P` have to be replaced by the disk and partition number, for sample: `(hd0,1)`.

If you want to add a custom iPXE script, just save a *plain-text* file containing the script somewhere on the disk (let's say in the `/boot/bootscript.ipxe` file), then add the line `initrd /boot/bootscript.ipxe` after the `kernel` entry in the configuration file:

```
menuentry iPXE {
  set root=(hd#D,#P)
```

```
    linux16 /boot/ipxe.lkrn
    initrd /boot/bootscript.ipxe
  }
```

Further information are available on iPXE's website at `http://ipxe.org/embed`.

3.3 GRUB2 disk

Note: In this part, we will only explain how to generate GRUB disks with GNU/GRUB version 2.

The software is packaged on most of Linux distros in the package *grub2*, you can also download the software at `http://www.gnu.org/software/grub/grub-download.html`.

In recents version of GRUB 2, it's possible to generate GRUB NBPs with the `grub-mkimage` utility. To do so the disk image have to be generated with the format `i386-pc-pxe` which is available on GRUB version 1.99.

The idea here is to generate a NBP GRUB disk image, let's say `grubpxe.0`. This file has to be loaded at boot time by the PXE-compatible NIC of nodes so it's necessary to configure the DHCP server to make this file be loaded at boot time (see section 1.4).

In order to be able to generate the disk image, GRUB 2 have to be installed on your machine (or at least, you should have compiled it).

To generate the GRUB disk image, you'll have to use the command:

```
grub-mkimage --format=i386-pc-pxe --output=grubpxe.0 \
  --prefix='(pxe)' MODULES
```

Where `MODULES` is a list of GRUB modules that you want to be embeded in the disk image. You can get a list of available modules in the GRUB modules directory (generally `/usr/lib/grub/` or `/boot/grub` depending on your system). At least you should include `pxe` (and `pxecmd` with GRUB 1.99).

For sample, if you want your image to be able to boot a Linux OS that is installed on an EXT2 partition of an MS-DOS partitioned hard disk, you'll have to include the modules `biosdisk`, `part_msdos`, `ext2` and `linux`.

Here is a list of modules that will make your image able to boot most generic Operating Systems:

```
pxe                             biosdisk
pxecmd   # GRUB 1.99 only      boot
pxechain # GRUB 2.0 only       configfile
net # GRUB 2.0 only            linux
tftp # GRUB 2.0 only           bsd
http # GRUB 2.0 only           lvm
```

```
multiboot                       hfsplus
part_apple                      iso9660
part_bsd                        ntfs
part_gpt                        reiserfs
part_msdos                      tar
part_sun                        udf
cpio                            ufs1
ext2                            ufs2
fat                             xfs
hfs                             zfs
```

If you want to embed a configuration script in your GRUB disk NBP (what will be pretty useful in the section 4), you can generate it with the command:

```
grub-mkimage --format=i386-pc-pxe --output=grubpxe.0 \
  --prefix='(pxe)' --config=grub.cfg MODULES
```

Where `grub.cfg` is a script containing GRUB 2 instructions.

4 Boot nodes following a node-specific profile

4.1 PXElinux/GPXElinux (core feature)

The NBP first try to download the profile from the file `pxelinux.cfg/MAC_ADDRESS` in the repository (`MAC_ADDRESS` is in lower case with `-` as separating token and prefixed with the hardware type (ARP type code)). If this file does not exist, it will try to download the file `pxelinux.cfg/HEXA_IP_ADDRESS` where `HEXA_IP_ADDRESS` is an hexadecimal representation of the IP address of the node (alpha digits are in upper case). If this file is not present, it will try to download truncated version of this filename and the fallback on the `pxelinux.cfg/default` file.

For example, if the MAC address of the node is `88:99:AA:BB:CC:DD` (with ARP type 1) and it's IP address is `192.0.2.91`, it will try to download profiles files in this specific order (unless one of this file exists in the repository):

```
pxelinux.cfg/01-88-99-aa-bb-cc-dd
pxelinux.cfg/C000025B
pxelinux.cfg/C000025
pxelinux.cfg/C00002
pxelinux.cfg/C0000
pxelinux.cfg/C000
pxelinux.cfg/C00
pxelinux.cfg/C0
```

```
pxelinux.cfg/C
pxelinux.cfg/default
```

It's also possible to custom the directory where the profiles are downloaded, the way the NBP download the profiles, etc... using some DHCP PXElinux-specific options:

- `Option 208` must be set to F1:00:74:7E for PXElinux to recognize any special DHCP options whatsoever.
- `Option 209` specifies the PXElinux configuration file name.
- `Option 210` specifies the PXElinux common path prefix, instead of deriving it from the boot file name.
- `Option 211` specifies, in seconds, the time to wait before reboot in the event of TFTP failure (0 means wait "forever")

For more information you can refer to the SYSLINUX wiki documentation at `http://www.syslinux.org/wiki/index.php/PXELINUX`.

4.2 iPXE (custom feature)

Booting the nodes following some instruction stored on a network-shared file is not a core feature of iPXE but there is a workaround to make it boot this way.

The idea is to embed a script file in your iPXE NBP/ROM/Kernel (see section 3.2 that will tell it to download a profile file from the network and boot the node following the instructions of this file.

If you want the iPXE NBP/ROM/Kernel to download the script file (profile) from TFTP with a profile filename based on node's IP address, the embeded iPXE script should look like something like this:

```
#!ipxe
dhcp
chain /ipxe.cfg/${net0/ip}
```

In this case, if the DHCP server gives the address `10.0.1.2` to the node that is bootimg, the iPXE NBP/ROM/Kernel will make the node download the script file `/ipxe.cfg/10.0.1.2` on the TFTP server and execute it (the TFTP server's address is given by DHCP). The script file `/ipxe.cfg/10.0.1.2` should contain the iPXE instructions to make the node boot.

Another example, if you want the iPXE NBP/ROM/Kernel to download the script file from HTTP with a profile filename based on node's MAC address, the embeded iPXE script should look like something like this:

```
#!ipxe
dhcp
chain http://bootserver.lan/ipxe.cfg/${net0/mac}
```

In this case, if the MAC address of the node's first NIC is `52:54:00:12:34`, the iPXE NBP/ROM/Kernel will make the node download the script file `/ipxe.cfg/52:54:00:12:34` on the HTTP server `bootserver.lan` and then execute it. If you want the filename to be formated as `52-54-00-12-34`, you can use `${net0/mac:hexhyp}` instead of `${net0/mac}`.

Further information about the iPXE commands and the variables it exports are available at `http://ipxe.org/cmd` and `http://ipxe.org/cfg`.

iPXE will also read different DHCP options (that you can customize in you DHCP server's configuration) that will change it's default behavior. This options are listed on iPXE web page at `http://ipxe.org/howto/dhcpd`.

4.3 GRUB2 disk (custom feature)

Booting the nodes following some instruction stored on a network-shared file is not a core feature of GRUB's NBP but there is a workaround to make it boot this way.

The idea is to embed a configuration file in your GRUB disk (see section 3.3 that will tell it to download a profile file from the network and boot the node following the instructions of this GRUB configuration file.

If you want the GRUB disk to download the configuration file from TFTP with a profile filename based on node's IP address, the embeded configuration file should look like something like this:

```
configfile (pxe)/grub.cfg/$net_pxe_ip
```

In this case, if the DHCP server gives the address `10.0.1.2` to the node that is bootimg, the GRUB disk will make the node download the configuration file `/grub.cfg/10.0.1.2` on the TFTP server and execute it (the TFTP server's address is given by DHCP). The configuration file `/grub.cfg/10.0.1.2` should contain the GRUB instructions to make the node boot.

Another example, if you want the GRUB disk to download the configuration file from HTTP with a profile filename based on node's MAC address, the embeded configuration file should look like something like this:

```
configfile (http,bootserver.lan)/grub.cfg/$net_pxe_mac
```

In this case, if the MAC address of the node's first NIC is `52:54:00:12:34`, the GRUB disk will make the node download the configuration file `/grub.cfg/52:54:00:12:34` with HTTP on the server `bootserver.lan` and then execute it (the DHCP server should be configured correctly if you want to use domain names, it's recommended to specify the HTTP server by IP address).

The GRUB NBP will also read different DHCP options (that you can customize in you DHCP server's configuration) and export them in variables.

- `Option 3`: *Router*, you can get the list of them with the command `net_ls_route`
- `Option 6`: *Domain Name Server*, you can get the list of them with the command `net_ls_dns`
- `Option 12`: *Host name*, exported in the variable `$net_pxe_hostname`
- `Option 15`: *Domain name*, exported in the variable `$net_pxe_domain`
- `Option 17`: *Root Path*, exported in the variable `$net_pxe_rootpath`
- `Option 18`: *Extensions Path*, exported in the variable `$net_pxe_extensionspath`

Further information about DHCP options ant the variables exported by GRUB are available at `http://www.gnu.org/software/grub/manual/grub.html#Network`.

5 Download and boot Operating System kernel

In this section we will describe PXE profiles (PXElinux profile, iPXE script, GRUB disk's configuration file) that can be used to make nodes boot a Linux kernel downloaded from the network.

In the following, we will try to boot the kernel `vmlinuz` and the initrd `initrd` that are stored in the directory `kernels/` of the TFTP, HTTP and FTP servers `bootserver.lan`. We will boot this kernel with the parameters `console=tty0 rw`.

5.1 PXElinux

Download from TFTP

```
DEFAULT networkboot
LABEL networkboot
  KERNEL /kernels/vmlinuz
  APPEND initrd=/kernels/initrd console=tty0 rw
```

Further information are available at `http://www.syslinux.org/wiki/index.php/SYSLINUX`.

5.2 GPXElinux

Download from HTTP

```
DEFAULT networkboot
LABEL networkboot
  KERNEL http://bootserver.lan/kernels/vmlinuz
  APPEND initrd=http://bootserver.lan/kernels/initrd console=tty0 rw
```

Download from FTP

```
DEFAULT networkboot
LABEL networkboot
  KERNEL ftp://bootserver.lan/kernels/vmlinuz
  APPEND initrd=ftp://bootserver.lan/kernels/initrd console=tty0 rw
```

Further information are available at `http://www.syslinux.org/wiki/index.php/SYSLINUX`.

5.3 iPXE

Download from TFTP

```
#!ipxe
kernel /kernels/vmlinuz console=tty0 rw
initrd /bootserver.lan/kernels/initrd
boot
```

Download from HTTP

```
#!ipxe
kernel http://bootserver.lan/kernels/vmlinuz console=tty0 rw
initrd http://bootserver.lan/bootserver.lan/kernels/initrd
boot
```

Download from FTP

```
#!ipxe
kernel ftp://bootserver.lan/kernels/vmlinuz console=tty0 rw
initrd ftp://bootserver.lan/bootserver.lan/kernels/initrd
boot
```

Note: To enable FTP support, you have to compile the iPXE NBP/ROM/Kernel with the `DOWNLOAD_PROTO_FTP` option (see `http://ipxe.org/err/3c092003`).

Alternative method

```
#!ipxe
initrd http://bootserver.lan/bootserver.lan/kernels/initrd
chain http://bootserver.lan/kernels/vmlinuz console=tty0 rw
```

Further information are available at `http://ipxe.org/scripting`.

5.4 GRUB2 disk

Download from TFTP

```
timeout=1
default=0
menuentry networkboot {
  linux (pxe)/kernels/vmlinuz console=tty0 rw
  initrd (pxe)/kernels/initrd
}
```

Note: A bug with TFTP files download (see `http://savannah.gnu.org/bugs/?36795`) was intruced in revision 4505 fixed in the revision 4548, be careful of the version of GRUB you are using.

Download from HTTP

```
timeout=1
default=0
menuentry networkboot {
  linux (http,bootserver.lan)/kernels/vmlinuz console=tty0 rw
  initrd (http,bootserver.lan)/kernels/initrd
}
```

Note: Since sometimes GRUB is not able to resolv domain names correctly (depending on your DHCP server's configuration) it's recommended to use the IP address of the HTTP server.

Alternative method

```
timeout=1
default=0
menuentry networkboot {
  set root=(http,bootserver.lan)
  linux /kernels/vmlinuz console=tty0 rw
  initrd /kernels/initrd
}
```

Further information are available at `http://www.gnu.org/software/grub/manual/grub.html`.

6 Boot local Operating System (from hard disk)

In this section we will describe PXE profiles (PXElinux profile, iPXE script, GRUB disk's configuration file) that can be used to make nodes boot an Operating System that is installed on hard disk.

In this sample, we will boot a Linux system that is installed on the third partition of the first hard disk. The kernel and initrd of the system are respectively located at `/boot/vmlinuz` and `/boot/initrd` in the EXT2 partition. We will boot the kernel with the options `console=tty0 rw`.

6.1 chain.c32 COMBOOT

Since PXElinux/GPXElinux and iPXE do not know filesystems (EXT3, FAT, ...) it's not possible to load and boot a kernel from the local hard disk of a node with this methods.

However it's possible to use a COMBOOT that will relay the boot procedure to a bootloader that is installed on the partition that we want to boot.

The COMBOOT `chain.c32` that is provided in the *syslinux* package/sources (see section 3.1) can be used to perform such kind of operation.

In the following we will see how to make `chain.c32` chain on the bootloader (such as GRUB) that is installed on the third partition of the first disk with different NBPs. The COMBOOT have to be present in the PXE repository (TFTP/HTTP).

6.1.1 PXElinux/GPXElinux

```
DEFAULT localboot
LABEL localboot
  KERNEL /chain.c32
  APPEND hd0 3
```

6.1.2 iPXE

```
#!ipxe
chain /chain.c32 hd0 3
```

Further information about the `chain.c32` COMBOOT are available on the *syslinux* wiki at `http://www.syslinux.org/wiki/index.php/Comboot/chain.c32`.

6.2 GRUB2 disk

First of all, in order to be able to boot a kernel that is installed on an EXT2 partition of an MS-DOS partitioned disk, the GRUB disk have to include the modules `biosdisk`, `part_msdos`, `ext2` and `linux` (see section 3.3).

The profile then look like:

```
timeout=1
default=0
menuentry localboot {
  set root=(hd0,3)
  linux /boot/vmlinuz console=tty0 rw
  initrd /boot/initrd
}
```

Depending on the modules you included in your GRUB disk, it can boot every system GRUB2 knows how to boot.

Another sample where we will ask to GRUB to chain on the bootloader that is installed on the partition (the same feature the `chain.c32` COMBOOT is providing):

```
timeout=1
default=0
menuentry localboot {
  set root=(hd0,3)
  chainloader +1
}
```

Further information about GRUB's boot methods are available at `http://www.gnu.org/software/grub/manual/grub.html#Booting`.

7 Chaining NBPs

Sometimes you may not want (or do not have the possibility) to change the NBP that is loaded by nodes depending on the DHCP server's configuration.

The idea in this section is to explain how NBP's can chainload each other in order to use a specific NBP regardless to the DHCP server's configuration (at least it should be configured to boot the nodes with one of the methods listed bellow).

For example, if your DHCP server is configured to make the nodes boot with PXElinux but you want them to boot with iPXE or a GRUB disk without having to modify the DHCP server configuration, you can make PXElinux load (or chainload) one of this other NBP.

In the following, `pxelinux.0` is a PXElinux NBP, `grubpxe.0` a GRUB NBP, `undionly.kpxe` an iPXE NBP and `ipxe.lkrn` an iPXE kernel file.

7.1 From PXElinux to GRUB2 disk

```
DEFAULT pxechain
LABEL pxechain
  PXE /grubpxe.0
```

7.2 From PXElinux to iPXE

```
DEFAULT pxechain
LABEL pxechain
  PXE /undionly.kpxe
```

7.3 From iPXE to PXElinux

```
#!ipxe
chain /pxelinux.0
```

7.4 From iPXE to GRUB2 disk

```
#!ipxe
chain /grubpxe.0
```

7.5 From GRUB2 disk to PXElinux

```
pxechainloader (pxe)/pxelinux.0
boot
```

Note: This feature contains a bug, the PXElinux NBP will be loaded but won't work, please refer to `http://lists.gnu.org/archive/html/grub-devel/2010-09/msg00049.html`.

7.6 From GRUB2 disk to iPXE

iPXE NBP

```
pxechainloader (pxe)/undionly.kpxe
boot
```

Note: This feature contains a bug, the iPXE NBP will be loaded but won't work, please refer to `http://lists.gnu.org/archive/html/grub-devel/2010-09/msg00049.html`.

iPXE Kernel

```
timeout=1
default=0
menuentry pxechain {
```

```
    set root=(pxe)
    linux16 /ipxe.lkrn
    initrd /bootscript.ipxe
  }
```

Where `bootscript.ipxe` is an *plain-text* file containing an iPXE script that will be run by the iPXE kernel.