PXE - Easy start using FATDOG Linux

Preface

In 2011, a Puppy contributor contributed a major addition to the setting up of a PXE server on a Puppy machine. This major enhancement provides “a single framework to work in” for starting the PXE server. He has made it almost instantly ready for LAN PCs to boot over the LAN directly from this server.

This document intends to be consistent with how someone can, quickly, go about getting his network operational such that a LAN PC can boot for a LAN server. And, it intends to be strongly consistent with all of the contributors contributions in the area; namely, PXE on Puppy Linuxes.

This manual does NOT intend to dwell on the particulars of architecture or structure. It simply provides some simple instructions on how to accomplish this.

This document is as accurate as one could make it. It may have inaccuracies, misspelling, and as such, you are instructed to use at your own risk. No contributor to this document’s generation is responsible in ANY WAY, with the outcome that can be achieved by its use. Even though much effort has gone into making this document accurate and straightforward; you are cautioned and advised to **use this document at your risk**.

Table of Contents

Introduction

For over 20 years, the ability to boot a PC from another PC has been available. There have been several technologies employed to do so. IBM-Intel introduced a spec and a technology to make this very easy. That spec, today, is known as PXE. This technology has been employed in EVERY motherboard that is currently manufactured which has on-board LAN connection(s) built-in. It is commonly known as PXE, and it supports booting the motherboard's PC from another source on your LAN. Thus, that motherboard does NOT require **any** local Operating System like Windows, Linux, Unix, OS2, BeOS**,** ReactOS, etc. In essence, it will get its Operating system over the LAN instead of its CD/DVD or hard-drive(HDD), or USB. The place, where its gets its stuff to boot and run, is from a place on your LAN called a "PXE server"; sometimes also called a "NetBoot server".

The way this works for the PC that is not the server is that:

1. it will power-on, its BIOS (Mac users: OpenFirmware) will tell the PC to check the LAN, first, for booting instructions
2. the PXE server will give the PC everything it needs to run
3. once booted, the PC would operate on its own without ANY further need/use of the PXE server.

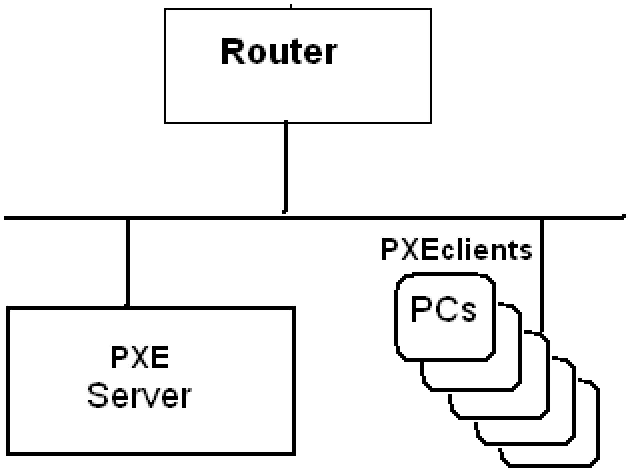
This offers enormous benefits to any LAN and certainly those for home, classroom(s), etc.

This is a brief instructional document that is used to setup such an environment which allows a "PXE PC" (also known as "PXE client") to boot from a "PXE server".

Objective

There are many scenarios for which Netbooting is useful. Once example is illustrated here: To setup a class room so that student PCs can boot Puppy Linux. Each student's PC will simply be started without having to install an OS on any PC. Booting a LiveCD on only one PC, acting as a PXE server, is enough for all PCs in the classroom to start and run Puppy, reliably.

The PXE server, when started, will start its own DHCP server to service booting for student PCs. This is necessary because for the "PXE PC" to boot successfully, the DHCP server use for Netbooting must provide additional information which isn't usually provided by standard DHCP server.



For this particular scenario, another "real" DHCP server is assumed to already exist on the network. This "real DHCP" server can be the "corporate DHCP server", or, it can be the DHCP services provided by many of the home routers. Two DHCP servers usually cannot co-exist, but with very clever configuration, this can be done - the Netboot DHCP server will be used during netbooting, and the standard "real" DHCP server will be used once the Puppy Linux has been "netbooted" up and is running.

The PXE server will also start a service which will allow PCs on the LAN to find the libraries they can use to boot their PCs over the LAN without the use of any media, local, on the booting PCs.

Assumption: We will ONLY address using wired ethernet devices in this document.

Preparation

Hardware

1. ***For the Server***
   a. Puppy LiveCD/LiveDVD
   b. 512MB RAM
   c. LAN
   d. crossover cable for isolated testing using only 1 PC and a PXE server (only required if there are problems, later)
2. **For the LAN PC capable of PXE** via its BIOS
   (Mac users: Use the OpenFirmware)
   a. 512MB RAM
   b. LAN via motherboard (not an adapter in a PCI slot)603600

Note 1: These 2 items will allow a minimal setup for PXE server and PXEclient

Software

1. ***For the Server***
   a. Two additional software pieces (PETs) for Puppy Linux
      i. dnsmasq - comes pre-configured for your use
      ii. netboot script - contains scripts and menu item
   b. The ISO for the Puppy derivative you want to start on the PXEclients
   c. A Puppy initramfs customized for net boot (aka “humongous initrd”) which will be built by using one of the netboot scripts

2. ***For the LAN PC capable of PXE***
   a. No software is used for this PC
   b. “nada”
   c. “zilch”
   d. “nothing”

Implementation - Basic for FATDOG

PXE server

Steps

The general steps to implement a PXE server is as follows. It prepares the programs and files required for this particular setup so that it can provide all necessary items that a PXEclient requires to boot.

Setup PXE for 1st-time Users - FATDOG (64bit) Version

Implementation Steps/Instructions

1. Go and install dnsmasq PET (you only need to do this once)
   For Fatdog64 - get it from [here]

2. Go and install the netboot script PET from [here] (you only need to do this once)

3. Download and save a Puppy Linux ISO you'll want to use for PXEclient(s) to boot. Save this somewhere - preferably outside your pup-save directory as ISO files can be quite large (100- 150MB, depending on which puppy you choose)

4. Run mknetboot.sh
   a. From console - open console/terminal, then type "mknetboot.sh" followed by Enter. Follow the prompts.
   b. From menu - open Menu, choose Utility, Gexec (Execute a Command). When a dialog opens, type "mknetboot.sh" and click "OK". Then follow the prompts.

5. Start/restart the PXE server's service by clicking menu > Network > Netboot Server

Important Note 2: When you have achieved success in your use of your PXE server, **you are advised to move the /tmp/netboot directory to a permanet location.** This folder contains the "real" files used during PXEclient booting. Its your choice where to save this. It can be saved to a local drive on your server or into a folder under the /root folder. **This is required iff** you want your setup efforts to survive over a reboot. Otherwise, the files that were built for your PXEclient to boot will be lost, even, when you do a "session save" at shutdown. This is because NOTHING in /tmp folder is ever saved over a reboot ("session save" of reboot DOES CAPTURE the contents of the /root folder.And, there are other choices you can make for a permanent location.)

potential replacement steps/instructions [you can use this or the above - up to you]

1. Install the appropriate DNSMASQ.PET

- For FATDOG users, download from here (64bit dnsmasq.pet)
- Check that dnsmasq is installed in your system's path by opening a terminal/console to the desktop and typing "dnsmasq -d". If you get a "command not found" response, report this as a DNSMASQ.PET bug.

2. Install the netboot-server.pet

- Same PET for any Puppy version, download from here
- The PET's installation creates the /root/tftpboot directory structure. It also adds gpxelinux.0 (from syslinux-4.03) and the "default" files into their proper locations.

3. Using the Downloaded ISO you want your PXEclients to boot up, then: From the ISO, extract the files necessary for PXEclient booting into the /root/tftpboot folder's chain

- After downloading, open a terminal window and type "mknetboot.sh". This script will led you to a chooser window where you must tell him where the ISO you downloaded is located. It will them, build the remaining files needed for PXEclient to boot. It also properly places them where the PXE client can find them when you power-on that PC.
- From step 2, above, the running of thIs script will symlink the proper vmlinuz and initrd.gz into /root/tftpboot/pxelinux.cfg directory so that any PXEclient PC gets all necessary files to boot to a Puppy desktop when powered on.

4. Start/Restart the Netboot-Server by clicking menu>Network>Netboot-Server

5. Power-on a PXE capable PC. It will boot over the LAN

Note 3: This applies to anyone using 32bit Puppy ISOs. The latest puppies of today (namely, Wary 5.0, Lupu 5.2, etc.), have a bug in their init script, causing PXEclient failure during booting this class of PUPs. In order to correct this, you may need to edit a change in the "default" file. While it's being fixed, a workaround is to edit the PXE server's "default" file and add "PDEV1=rootfs" to the line that contains the word "append". Assuming you used the steps above, the file's path is **/root/tftpboot/pxelinux.cfg/default.** Earlier version of puppies, as well as future version of puppies, will work fine (the bug has been reported and fixed for future releases).

This completes server preparation. This machine is now waiting to "service" any PXEclient needing to boot.

PXE Client

To have your PC to start as a PXEclient, you must tell it use its PXE function. This is a BIOS controlled feature and can be utilized in one of 2 ways:

1. Automatically
2. Use the PC's startup "hotkey"

Automatically: You have already entered your BIOS, and set it up so that it will look to the "Network", first, for booting instructions. When this is saved, the PC, when it boots, will start its PXE function.

OR

Hotkey: Many PCs provide a "hotkey" option at boot so that you can instruct the computer to start from an alternate location ie. CDROM, USB, 2nd HDD, or LAN. If you select LAN, the PC will start its PXE function.

Once you are prepared and have performed one of the above 2 steps,
Restart the PC; Stand back!; Smile when it "barks"

Caution:

All Puppy distros have a *.sfs that is shipped with the Puppy ISO you've downloaded. If any of those ISO "sfs" files exists on the PXEclient's HDD/USB/CD/DVD devices, it "may" affect your PXEclient booting.

For example, if there exists a lupu-511.sfs anywhere, in any device on your PC you are trying to get started using PXE, Puppy will uses that one, and will ignore the one which is present in the initramfs that resides on the PXE server! The one which is saved on the PXE server is the one the PXEclient is suppose to use. So when he uses the lupu-511.sfs on your local PC, problems can arise.

It's useless to go further if you don't have PXEclient boot success at this stage. You should retrace the steps outlined to get the PXEserver setup so that it boots your Puppy PXEclient. Otherwise, you will have to troubleshoot the PXE system's problems.

One sucessful PXE setup,using FATDOG, went as follows

- The above steps have been tested and found to work, flawlessly using Puppy FATDOG as a PXE server.
- The PXE server manipulated the ISO of a LightHouse 5.02 Puppy linux for use by any PXEclients.
- Several 32bit PXEclients were tested.

- Each PXEclient booted to a Lighthouse desktop, over the LAN, from the PXE server.
- Once booted, the PXEclients disconnected from the PXE server and operated on their own.

One should expect that these steps would work the same when using a 32bit Puppy as a PXE server, too.

Files Layout after Basic Implementation

When netboot script is installed, it will create the following minimal directory hierarchy required for Netbooting to work. If you want to change the default settings, then you need to know this structure. The Netboot-server script creates this structure, identically, for all Puppies.

Netboot directory structure and their explanations.
/root/tftpboot/ <== the root directory for Netbooting (Only 4 files are needed in this directory for a PXEclient to boot.) /root/tftpboot/pxelinux.0 <== Netboot loader, comes from the pet /root/tftpboot/pxelinux.cfg/default <== Netboot config file, comes from the pet. /root/tftpboot/pxelinux.cfg/vmlinuz <== the linux kernel for use by PXE client, created by mknetboot.sh /root/tftpboot/pxelinux.cfg/initrd.gz <== the huge initrd for use by PXE client, created by mknetboot.sh

Implementation #2 - Get fancy a little

All right, so the basic works. PXE PC boots straight to puppy, no question asked. For a little fancy, how do we get to show the menus that is normally shown when one boots from LiveCD?

Steps

All you need to do to show the menus is to follow these steps:

1. Open the ISO file that you have used with mknetboot.sh which created that huge initrd.
2. **Copy everything** on that ISO Copy everything, files and folders, to /root/tftpboot/pxelinux.cfg/, **except the following**, that I mention below:
    a. Don’t copy vmlinuz
    b. Don’t copy initrd.gz
    c. Don’t copy any files that ends with .sfs (*.sfs)
    d. Don’t copy boot.cat and isolinux.bin
3. Rename the /root/tftpboot/pxelinux.cfg/default to something else (don’t delete it - you may need it for reference)
4. Edit /root/tftpboot/pxelinux.cfg/isolinux.cfg (that you’ve just copied from the ISO), find anything that says “pmedia=cd” and delete it. Don’t delete the entire line, just delete the words “pmedia=cd”.
5. Rename /root/tftpboot/pxelinux.cfg/isolinux.cfg to /root/tftpboot/pxelinux.cfg/default
6. You’re system is ready to go.

Note: You can do the above while the Netboot server is still running.

Troubleshooting Tips

Most often problem: The latest puppies of today (namely, Wary 5.0, Lupu 5.2), have a bug in their init script, causing PXEclient failure during booting this class of PUPs. While it's being fixed, a workaround is to edit the PXE server's "default" file and add "PDEV1=rootfs" to the line that contains the word "append".

If you retraced you steps and are still experiencing difficulties, you may want to setup an Isolated Test environment.

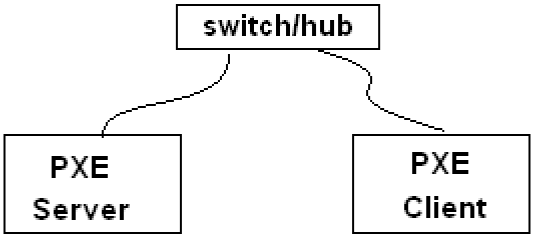
Setup an “Isolated Test” Environment

This provides a sure-fired way of knowing whether your PXE server is setup properly so that a PXEclient can boot from it. In this Isolated Test, we will remove both the server and the PXEcient from your network LAN. Then, you willconnect, only, these 2 PCs together for this test. These will be the only 2 PCs in this isolated environment. This environment can be set up in one of two ways:

- Easiest method - you can use a separate ethernet switch/hub; not your main LAN
- Little harder - you can use a cross-over cable in place of a switch

In the ‘easiest method’:

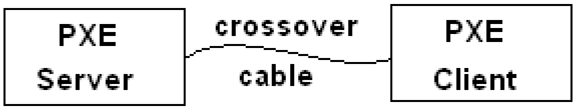
1. connect the PXEserver’s ethernet LAN cable to any port on the switch/hub and connect the PXEclient’s LAN cable to the next port on that same switch/hub.



2. boot up the Puppy PXEserver
3. On the server desktop, click menu > Network > Netboot Server to get it restarted
4. Then we are going to power-on the PXEclient to see if it connects

In the ‘little harder’

1. using a cross-over cable, we plug one end of cable into the PXEserver’s and we plug the other end of the cable into the PXEclient



2. boot up the Puppy PXEserver
3. Assign the following network personality
4. On the server desktop, click menu > Network > Netboot Server to get it restarted
5. Then we are going to power-on the PXEclient to see if it connects

Note: On the PXEserver, we MAY need to set a static address, mask, and gateway for these units on the isolated network to “see” each other so that the PXEclient can boot.

Is your ISO OK?

To insure that your ISO is NOT corrupt, burn it to a CD. Take it to a PC, and see if it boots to the desktop without a “kernal panic” or some other error.

PXEclient Stopped

...job control turned off

You have probably setup one of the 32bit Pups that has a bug in the init script. to correct, **see note 3, above**.

...File Not Found

You may not have set your TFTP folder up correctly.
Try following the setup steps again for your ISO.

...Invalid or Corrupt kernel image

The build of your PXE files created a corrupt file.
Download the ISO (check it against its MD5 checksum) and follow the setup steps for your ISO.

...Kernel Panic - not syncing: VFS Unable to ….

You may have setup one of the 32bit Pups that has a bug in the init script. to correct
see note 3, above.
