PXE: Installing Slackware over the network

When the time comes to install Slackware on your computer, you have a limited number of options regarding the location of your Slackware packages. Either you install them from the (un)official Slackware CDROM or DVD, or you copy them to a pre-existing hard disk partition before starting the installation procedure, or else you fetch the packages from a NFS server, FTP or HTTP server.

The number of available options for booting your Slackware installer is similarly limited: either you boot your computer from the bootable first CDROM of the Slackware CD set, or from the DVD, or using a USB stick.
There is even loadlin, the DOS based Linux starter, but lets not concern ourselves with the past today. Slackware 12.0 abandoned the floppy boot altogether.

But what if your PC is lacking a CDROM drive? Brands of PC's are on the market today (ultra-portable laptops for instance) that are unable to install Slackware the traditional way. However, these machines are commonly equipped with network peripherals, like bluetooth, wireless and wired network cards. How to solve this dilemma? Buy an external CD drive?

Well, there is another way of booting your computer that the Slackware installer supports. That is the *network boot*. Network boot, or “PXE boot”, requires support from your computer's network card and BIOS. Also, instead of installing packages from a Slackware CDROM set, you will need a network server that can instruct your computer how to fetch those packages from the network.

In this README, I will show you how to perform an installation that uses the network as the carrier medium, with a server on the local network that holds the boot kernel and the root filesystem (which contains the setup program), and also has all the Slackware packages. This means, there is no need for a floppy or CDROM drive.
Be warned: setting it all up is not trivial, and you need more than a beginner's level of Linux knowledge, but this text and the accompanying example scripts in the last section should get you up and running even if you do not completely understand what is going on 🙂

I will describe how to setup a server with the proper software, and how to use some of the files on the Slackware CDROM so that PXE-boot becomes a supported installation method for Slackware Linux. Alternatively, read the README_PXE.TXT file on your Slackware CDROM.

PXE

The commonly used method of booting a computer over the network is called PXE or **P**reboot E**x**ecution **E**nvironment. If you want your computer to boot using PXE, it needs a network card with PXE-capable firmware, and a BIOS that supports network boot. Most modern network cards (and computers) sold on the market today support this. When a computer boots from the network , it is the network card that downloads the bootloader, kernel and a filesystem - any Operating System that might already be installed on the computer will be untouched.
You can just as well boot a diskless computer using PXE - in fact this is how Thin Clients and the Linux Terminal Server Project work.

Of course, the other end of the network needs our attention, too. A *PXE server* needs to be available

on the local network. The PXE firmware in your computer's network card will contact this server in order to fetch some kind of bootable program code and bootstrap itself. What happens *after* the computer boots has no longer anything to do with the PXE boot stage, it is the bootstrapping process we're interested in.

We will cover the requirements for such a server in one of the next paragraphs.

The Slackware network installation process will roughly be as follows:

- You start the computer that is going to receive Slackware;
- On startup, you make sure you select *network boot* in the BIOS startup - either by activating a `custom startup sequence`, by entering a `boot menu` by pressing a function key like F12, or else by preselecting `network boot` as first option in the BIOS.
- When the computer boots, the network card activates its PXE code and tries to contact a PXE server. When such a server exists on the LAN, it will tell the card where to download a piece of bootable code, an Operating System kernel (the Slackware Linux kernel) and an initial ramdisk (aka initrd - the compressed root-filesystem image where the setup program, libraries and kernel modules are stored). You will see a page full of mumbo-jumbo as the card broadcasts on the LAN, and probes for possible candidate configurations to download;
- If a willing PXE server was found, your computer's network card will then download a kernel and initrd, boot the Linux kernel, unpack the initrd into a ramdisk and start the Slackware installer's initialization sequence. This is where you'll be in familiar territory again, since this is exactly what happens if you had booted from a CDROM or a floppy. But the fun is not over...
- Since we booted the computer using code that did not originate from our computer, we will have to fetch the remainder of the data - the Slackware packages - from the network as well. It's just that the freshly booted Linux kernel has no idea how it came to be running on the computer: you will have to initialize the network all over again. The network card's PXE firmware has done it's job and is no longer in the picture. So:
- We need to load a kernel driver for our network card and locate a network server that holds the Slackware package tree. Currently, NFS, FTP or HTTP installs are supported means of getting to your network data. The Slackware installer will use udev to configure your network card. If your network card is not supported by any of the available drivers, you're out of luck and will have to re-evaluate your options.
- From here on, installation proceeds as usual, under the condition that you select *Install from NFS (Network File System)* or *Install from FTP/HTTP server* as the source of the Slackware packages.

Workstation requirements

As stated before, the requirements for the computer you want to install Slackware on, are as follows:

- network card (non-wireless) with *PXE* firmware, supported by Slackware
- PC BIOS allowing to select *network boot*

No other requirements have to be met for a network install, other than those you'd already have to meet in order to be able to install and run Slackware.

Server requirements

This is the interesting part (well in my opinion at least - many people consider this as a dark art).
A PXE Server is really a mix of several components. We need

- A service that understands the *BOOTP* protocol. BOOTP is a network protocol somewhat like DHCP, and it is used by the PXE firmware to broadcast on the network its desire to find a suitable server to download the bootstrap code from. The ISC DHCP Server which comes as part of Slackware fulfills this requirement, since it supports BOOTP as well as DHCP.
- A download service for the bootstrap code. A *TFTP* (trivial file transfer protocol) server is needed for this. Slackware ships with an implementation of a TFTP server called tftpd-hpa which does what we need.
- And for the Slackware installer, a NFS, FTP or HTTP server is required because we must perform a network install. We can use Slackware's stock NFS/FTP/HTTP server for that. In the example below, we will configure a NFS server.

Implementation of the services

We'll look at how to set up the DHCP, TFTP and NFS services on a Slackware computer so that they work together as a PXE server.

For ease of instruction, I will make a number of assumptions. These assumptions are reflected in IP addresses and address ranges that I use in my examples, in the names of directories, computers and network domains. This means that if **you** use the examples in this article, you should make certain that you replace all occurrences of these specifics with values that apply to your own network.

- Our example network uses IP addresses in the range of **`192.168.0.0`** to **`192.168.0.254`**. This is equivalent to a network range **192.168.0.0/24** or **192.168.0.0/255.255.255.0**.
- Our network server will have the IP address of **192.168.0.1** and the default gateway is **192.168.0.10**. Server and gateway can be (but do not need to be) the same physical machine.
- The IP address range that the DHCP server will use to lease to DHCP/BOOTP enabled computers is **`192.168.0.50`** to **`192.168.0.100`**.
- The DNS domain will be “**`my.lan`**”.
- The server will run all required services, i.e. acts as the LAN's **DNS,DHCP, TFTP and NFS** server. If you decide to separate DHCP and TFTP services onto two different servers (it does not matter where the NFS server runs), I will add a comment on what you should take care of in the DHCP section that comes next.
- Directories are used as follows:
  - Top level of the complete Slackware 12.2 directory tree (excluding the source code if you're short on disk space) is **`/mirror/Slackware/slackware-12.2`**
  - The directory where we store the boot files for the TFTP server is **`/tftpboot/slackware-12.2`**

DHCP

You probably already have a DHCP server running on your network. You can try and modify its configuration so that it will do want we want, or if that is impossible (for instance because the DHCP server is running on your DSL/Cable router) you could consider disabling that and setting up a Slackware DHCP server for your LAN with much enhanced functionality.

Slackware includes the *ISC DHCP* server package (dhcpd). Two example `/etc/dhcpd.conf` configuration files for this DHCP Server are included in the last section of the article.

If you don't want to be bothered with fancy configurations but want a quick solution that will just work for your network, use the first (simple) example `/etc/dhcpd.conf` configuration file as well as the provided `/etc/rc.d/rc.dhcpd` start script and you'll be up and running in minutes. It requires *no* editing of files, the examples will work out of the box.

If you know what you're doing and understand (more or less) how the DHCP server works, you can have a look at the second, more complex, `/etc/dhcpd.conf` example which has more features and offers control over what computers are allowed to do a network boot.
The rest of this chapter deals with the setup of a *complex* DHCP configuration.

By default, we should no allow network boots in our network (which is safer of course - imagine a computer that does an un-intended network boot and suddenly finds itself running the Slackware installer!). In the `/etc/dhcpd.conf` configuration file, we add a *group* section were we can add those computers that we allow as network boot clients; the typical *host* statement for a computer looks like this

```
host t43 {
    hardware ethernet 00:12:34:56:78:9a;
    fixed-address 192.168.0.3;
}
```

Nothing spectacular; a computer is defined by the network card's hardware address (MAC address) and we let the DHCP Server always assign it the same IP address. The boot-specific parameters are all contained in the *group block* and look like this:

```
group {
  allow bootp;
  next-server 192.168.0.1;
  use-host-decl-names on;
  if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
    filename "/slackware-12.2/pxelinux.0";
  }

  host ABC {
    .....
  }

  host XYZ {
    .....
  }
```

```
}
```

This makes the DHCP server recognize network boot clients that use PXE and serves them the PXElinux boot loader `/slackware-12.2/pxelinux.0`. What this boot loader does will be explained further down the article.

The `next-server` parameter contains the IP adddress of the TFTP server. This will often be identical to the DHCP server's IP address, but if you have a TFTP server that is running on a different IP address than the DHCP server (i.e. they run on separate servers) you will have to add the remote IP address instead, like this (assuming the TFTP server is running on IP address 192.168.0.254):

```
next-server 192.168.0.254;
```

If you are running a version of ISC dhcpd that is >= 3.0.3 , then the addition of a `next-server <ipaddress>;` line is **mandatory**. For older releases this was only needed if the TFTP and DHCP Servers actually had different addresses.
If you fail to set the `next-server` address, the *siaddr* field in the data returned to the client is set to zero where in the past it would default to the DHCP server's own IP address (which often happened to be the IP address of the TFTP server as well). The PXE client uses the *siaddr* field to determine the IP address of the TFTP server and so the PXE booting will stall at the point of looking for a TFTP server.

If you are already using **dnsmasq** as your DNS/DHCP server, then the above instructions for the ISC DHCP server are not applicable to your setup. In that case, I have another Wiki article (which focuses on QEMU) for you where I have documented the required modifications to the dnsmasq server.

TFTP

The tftpd service is managed by inetd. Enable the line for tftpd in the file `/etc/inetd.conf`by removing the comment character at the beginning of the line:

```
tftp  dgram  udp  wait  root  /usr/sbin/in.tftpd  in.tftpd -v -s /tftpboot -
r blksize
```

and reload inetd:

```
/etc/rc.d/rc.inetd restart
```

We need to create the directory `/tftpboot` which will hold the bootstrap files that tftpd will serve:

```
mkdir /tftpboot
```

This directory is the root of a secure jail (the `-s /tftpboot` parameter in `/etc/inetd.conf`). The tftpd service is now configured and running. We just need to populate it's root directory, but I'll keep that for another paragraph.

NFS

For setting up a NFS server, I simply point you to another article in this Wiki: file- and printersharing on the local network.

- You need to export the directory where you keep your local copy of the Slackware packages, for instance by adding this to `/etc/exports`:

```
/mirror/Slackware       192.168.0.0/24(ro,sync,insecure,all_squash)
```

  in case your Slackware packages are located somewhere below `/mirror/Slackware` (like in our example network). The above line makes this directory tree available (read-only) to NFS clients in the local network defined by the IP address range `192.168.0.0/24`.

- If you had to add this to the `/etc/exports` file, you will need to restart the NFS server:

```
/etc/rc.d/rc.nfsd restart
```

For simplicity's sake: these are the steps if you just want this working and have no prior experience with NFS nor have such a server running at the moment (remember that pathnames/IP addresses are used that apply to our example network - adjust as needed):

- Create a file named `/etc/exports` with the following content:

```
/mirror/Slackware       192.168.0.0/24(ro,sync,insecure,all_squash)
```

- Use the directory mentioned in `/etc/exports` to copy (or move) the contents of your Slackware files to: The directory `/mirror/Slackware/slackware-12.2` should be the root of the Slackware tree, containing such files as the `ChangeLog.txt` and subdirectories like `slackware` and `kernels`.
- Make the NFS server startup script executable so that the NFS server will start on every boot:

```
chmod +x /etc/rc.d/rc.nfsd
```

- Start the NFS server (so you don't have to reboot already):

```
/etc/rc.d/rc.nfsd start
```

PXELinux configuration

The previous paragraph described in generic terms how to setup the TFTP service, but it did not tell you how to populate the TFTP directory structure so that network clients can request and obtain boot files.

PXELinux is much like isolinux, which is the bootloader that is used for the bootable Slackware CDROM #1. In fact, both programs are written by the same author and are available in Slackware via the

syslinux package.

The tftp directory structure

The usual way of installing Slackware from a NFS server repository, is to boot the Slackware CDROM, and let udev load the driver for your network card before starting `setup` to commence the installation procedure.
Now, with a boot from the network we are assuming there is *no CDROM* and *no floppy* drive available to us. So, we need to have all the drivers we need inside the root filesystem which the PXElinux bootloader downloads from the TFTP server. This root filesystem is wrapped in the `./isolinux/initrd.img` file (the initial ramdisk image). In fact, this image file is exactly the one which a CDROM installation uses as well. Slackware used to have separate `network.dsk` and `pcmcia.dsk` images but these have been folded into the `initrd.img` for ease of maintenance.

As you can see in the DHCP section, the DHCP server offers any interested PXE client (i.e. your computer's network card) the file `/slackware-12.2/pxelinux.0` - this file contains the bootable code that subsequently downloads and starts a Linux kernel, and downloads and extracts the root filesystem containing the setup program and everything else that we need.
This filename `/slackware-12.2/pxelinux.0` indicates a pathname relative to the root of the TFTP server. The PXE client will use the tftp protocol to fetch this bootloader. So this is what we do: create this directory `slackware-12.2` and copy the required files into it. First, the pxelinux bootloader itself:

```
mkdir /tftpboot/slackware-12.2
mkdir /tftpboot/slackware-12.2/pxelinux.cfg
cp /usr/share/syslinux/pxelinux.0 /tftpboot/slackware-12.2/
```

Also, we need the files from the Slackware CDROM that show the informative messages in the beginning. Assuming your mirror of the Slackware release can be found in /mirror/Slackware/slackware-12.2/ (change paths in the below commands if your location is different) :

```
cp /mirror/Slackware/slackware-12.2/isolinux/message.txt
/tftpboot/slackware-12.2/
cp /mirror/Slackware/slackware-12.2/isolinux/f2.txt
/tftpboot/slackware-12.2/
cp /mirror/Slackware/slackware-12.2/isolinux/f3.txt
/tftpboot/slackware-12.2/
```

And we need *all* the kernels that you can choose from after booting the installer:

```
cp -a /mirror/Slackware/slackware-12.2/kernels /tftpboot/slackware-12.2/
```

Lastly, we copy the initrd.img file (which contains the installer and the network/pcmcia kernel modules) as well as a pxelinux configuration file:

```
cp /mirror/Slackware/slackware-12.2/usb-and-pxe-
installers/pxelinux.cfg_default
/tftpboot/slackware-12.2/pxelinux.cfg/default
cp /mirror/Slackware/slackware-12.2/isolinux/initrd.img
```

```
/tftpboot/slackware-12.2/
```

Trying it out

You now have a fully configured PXE server. Try it out!
Take a computer that is able to do a network boot, start it, and watch it go through the motions of contacting the PXE server, downloading the PXE boot code and presenting you with the familiar Slackware installation screen! From there on you're on familiar grounds: choose a kernel, and off you go.

- Select a “NFS installation” once you get to the “SOURCE” dialog. You will need to supply a couple of values for IP Addresses and the NFS server directory. These are:

Your own IP Address (pick any unused)	192.168.0.111
Your netmask	255.255.255.0
The gateway	192.168.0.10
NFS server address	192.168.0.1
Slackware directory on the NFS server	/mirror/Slackware/slackware-12.2/slackware

- From this point onwards, the installation proceeds just as when the SOURCE would have been a CDROM.
- If you have configured a FTP or HTTP server with a Slackware package tree instead, the dialogs are similar.

Good luck!

Example configuration scripts

First example dhcpd.conf

A simple `/etc/dhcpd.conf` where all computers are allowed to boot from the network using PXE.

```
# dhcpd.conf
#
# Configuration file for ISC dhcpd
#

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;
ddns-update-style none;
```

```
# Allow bootp requests
allow bootp;

# Point to the TFTP server:
next-server 192.168.0.1;

# Default lease is 1 week (604800 sec.)
default-lease-time 604800;
# Max lease is 4 weeks (2419200 sec.)
max-lease-time 2419200;

subnet 192.168.0.0 netmask 255.255.255.0 {
    option domain-name "my.lan";
    option broadcast-address 192.168.0.255;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 192.168.0.1;
    option routers 192.168.0.10;
    range dynamic-bootp 192.168.0.50 192.168.0.100;
    use-host-decl-names on;
    if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
      filename "/slackware-12.2/pxelinux.0";
    }
}
```

Second example dhcpd.conf

A more advanced `/etc/dhcpd.conf` file for your DHCP server where you can specify exactly which computers are allowed to boot from the network using PXE (but you will have to collect their MAC addresses yourself and put them into separate *host{}* entries):

```
# dhcpd.conf
#
# Configuration file for ISC dhcpd
#

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;
ddns-update-style none;

# Ignore bootp requests:
ignore bootp;

# option definitions common to all configured networks...
option domain-name-servers 192.168.0.1;

subnet 192.168.0.0 netmask 255.255.255.0 {
```

```
    option domain-name "my.lan";
    option broadcast-address 192.168.0.255;
    option subnet-mask 255.255.255.0;
    option routers 192.168.0.10;
    # We reserve the range 192.168.0.1 to 192.168.0.49 for static IP
addresses
    pool {
      # Known clients (i.e. configured with a 'host' statement)
      # that request an IP address via DHCP
      range 192.168.0.50 192.168.0.100;
      # Default lease is 1 week (604800 sec.)
      default-lease-time 604800;
      # Max lease is 4 weeks (2419200 sec.)
      max-lease-time 2419200;
      deny unknown clients;
    }
    pool {
      # Guests
      range 192.168.0.150 192.168.0.200;
      # Default lease is 8 hours (28800 sec.)
      default-lease-time 28800;
      # Max lease is 24 hours (86400 sec.)
      max-lease-time 86400;
      deny known clients;
    }
}

# Hosts which require special configuration options can be listed in
# host statements. If no address is specified, the address will be
# allocated dynamically (if possible), but the host-specific information
# will still come from the host declaration.

# Fixed IP addresses can also be specified for hosts. These addresses
# should not also be listed as being available for dynamic assignment.
# Hosts for which fixed IP addresses have been specified can boot using
# BOOTP or DHCP. Hosts for which no fixed address is specified can only
# be booted with DHCP, unless there is an address range on the subnet
# to which a BOOTP client is connected which has the dynamic-bootp flag
# set.

# === Group definitions ========================================
# Define groups of computers that you want to give special attention.

group {
  # Non-PXE machines

  # Default lease is 1 week (604800 sec.)
  default-lease-time 604800;
  # Max lease is 2 weeks (1209600 sec.)
  max-lease-time 1209600;
```

```
  #host penguin {
  #  hardware ethernet xx:xx:xx:xx:xx:xx;
  #  fixed-address 192.168.0.2;
  #}
}

group {
  # PXEboot

  # Default lease is 1 day (86400 sec.)
  default-lease-time 86400;
  # Max lease is 2 days (172800 sec.)
  max-lease-time 172800;

  # Allow bootp requests for this group:
  allow bootp;

  # Point to the TFTP server (required parameter!):
  next-server 192.168.0.1;

  # If you want to log the boot process, you will need to configure
  # your logserver to allow logging from remote hosts.
  #option log-servers 192.168.0.1;

  use-host-decl-names on;

  if substring (option vendor-class-identifier, 0, 9) = "PXEClient" {
    filename "/slackware-12.2/pxelinux.0";
  }
  else if substring (option vendor-class-identifier, 0, 9) = "Etherboot" {
    filename "/slackware-12.2/kernels/hugesmp.s/bzImage";
  }

  host t43 {
    hardware ethernet yy:yy:yy:yy:yy:yy;
    fixed-address 192.168.0.3;
  }
}  # end of PXEboot group
```

RC script

A Slackware start/stop script for the DHCP server that you can save as `/etc/rc.d/rc.dhcpd`.

Don't forget to make the script executable:

```
chmod +x /etc/rc.d/rc.dhcpd
```

. You can add the following lines to `/etc/rc.d/rc.local` so that the DHCP service starts when your server boots:

```
if [ -x /etc/rc.d/rc.dhcpd ]; then
  # Start the DHCP server:
  /etc/rc.d/rc.dhcpd start
fi
```

```
#!/bin/sh
#
# /etc/rc.d/rc.dhcpd
#      This shell script takes care of starting and stopping
#      the ISC DHCPD service
#

# Put the command line options here that you want to pass to dhcpd:
DHCPD_OPTIONS="-q eth0"

[ -x /usr/sbin/dhcpd ] || exit 0

[ -f /etc/dhcpd.conf ] || exit 0

start() {
      # Start daemons.
      echo -n "Starting dhcpd:  /usr/sbin/dhcpd $DHCPD_OPTIONS "
      /usr/sbin/dhcpd $DHCPD_OPTIONS
      echo
}
stop() {
      # Stop daemons.
      echo -n "Shutting down dhcpd: "
      killall -TERM dhcpd
      echo
}
status() {
  PIDS=$(pidof dhcpd)
  if [ "$PIDS" == "" ]; then
    echo "dhcpd is not running!"
  else
    echo "dhcpd is running at pid(s) ${PIDS}."
  fi
}
restart() {
      stop
      start
}

# See how we were called.
case "$1" in
  start)
        start
```

```
        ;;
  stop)
        stop
        ;;
  restart)
        stop
        start
        ;;
  status)
        status
        ;;
  *)
        echo "Usage: $0 {start|stop|status|restart}"
        ;;
esac

exit 0
```