Diskless Bootstrap in OpenBSD 3.x for Intel Architectures

Vassilis Prevelakis (vp@drexel.edu)

Drexel University/CS

ABSTRACT

The ability to boot a machine over the network is useful for diskless or dataless machines, or as a temporary measure while repairing or re-installing filesystems on a local disk. This file provides a general description of the interactions between a client and its server when a client is booting over the network. The general description is followed by specific instructions for configuring a server for diskless i386 clients.

1. Before you start

Note that you need a boot loader that can load OpenBSD kernels from the network. One such loader is GRUB. You also need to apply some patches to your client kernel. Information about all this may be found in `<http://lists.gnu.org/archive/html/bug-grub/2001-03/msg00077.html>`

Note, however, that now you can compile the grub source natively under OpenBSD (i.e. you can ignore the note about getting grub compiled under Linux).†

2. Operation

When booting a system over the network, there are three phases of interaction between client and server:

1. The PROM (or stage-1 bootstrap) loads a boot program.
2. The boot program loads a kernel.
3. The kernel mounts (via NSF) the root filesystem and swap partition.

At the end of this sequence the system can operate in single-user mode. If the system continues the boot process and goes multi-user additional file systems may be mounted (possibly via NFS) as specified in the `/etc/fstab` file.

In phase 1, the PROM loads a boot program. PROM designs vary widely, so this phase is inherently machine-specific. Sun and Motorola machines use RARP to determine the client's IP address and then use TFTP to download a boot program from whoever sent the RARP reply. HP 300-series machines use the HP Remote Maintenance Protocol to download a boot program. Intel 386-class machines sometimes have a PROM on the network card (NIC) that uses the PXE protocol which is a minor modification to the DHCP protocol. Other machines may load a network boot program either from diskette or using a special PROM on the network card.

This instruction sheet assumes that your NIC supports the PXE protocol.

In phase 2, the boot program loads a kernel. Operation in this phase depends on the design of the boot program. The boot program:

2.1 Gets the client IP address using DHCP.

2.2 Gets the client name and server IP address by broadcasting an RPC / BOOTPARAMS / WHOAMI request with the client IP address.

† Actually, you could compile grub in previous OpenBSD releases, but it was more fiddly.

2.3 Gets the server path for this client's root using an RPC / BOOTPARAMS / GETFILE request with the client name.

2.4 Gets the root file handle by calling `mountd(8)` with the server path for the client root.

2.5 Gets the kernel file handle by calling NFS lookup on the root file handle.

2.6 Loads the kernel using NFS read calls on the kernel file handle.

2.7 Transfers control to the kernel entry point.

In phase 3, the kernel does NFS mounts for root and swap. The kernel repeats much of the work done by the boot program because there is no standard way for the boot program to pass the information it gathered on to the kernel. The procedure used by the kernel is as follows:

3.1 The kernel finds a boot server using the same procedure as described in steps 2.1 and 2.2 above.

3.2 The kernel gets the NFS file handle for root using the same procedure as described in steps 2.3 through 2.5 above.

3.3 The kernel calls the NFS `getattr` function to get the last-modified time of the root directory, and uses it to check the system clock.

3.4 If the kernel is configured for swap on NFS, it uses the same mechanism as for root, but uses the NFS `getattr` function to determine the size of the swap area.

3. Configuration

Before a client can boot over the network, its server must be configured correctly. This example will demonstrate how an i386 client might be configured − other clients should be similar.

Assuming the client's hostname is to be `myclient`,

1. Assign an IP address for myclient in your `/etc/hosts`, or DNS database:

```
192.197.96.12           myclient
```

2. Add an entry to `/etc/dhcpd.conf :`

```
host myclient {
    hardware ethernet 00:02:20:7:c5:c7;
    fixed-address 192.197.96.12;
    option host-name "myclient";
    filename "/pxegrub";
}
```

3. Ensure that `/etc/inetd.conf` is configured to run `tftpd(8)` in the directory `/tftpboot`.

4. Build an OpenBSD kernel that expects its root and swap partitions to be accessed via NFS. We will use the DISKLESS kernel configration file supplied with the OpenBSD 3.x distribution.

```
cd /sys/arch/i386/conf
config DISKLESS
cd ../compile/DISKLESS
make depend
make
cp -p bsd /tftpboot/bsd
```

5. Install a copy of the PXE boot loader (`pxegrub`) in the `/tftpboot` directory.

6. Add myclient to the bootparams database `/etc/bootparams`:

```
myclient  root=server:/export/myclient/root \
          swap=server:/export/myclient/swap
```

Make sure that the server runs the correct daemons. File `/etc/rc.conf.local` must contain at least the following lines:

```
dhcpd_flags="-q"
nfs_server=YES
bootparamd_flags=""
```

If any these lines are missing, you will have to reboot the server after you update the `/etc/rc.conf.local` file (unless you know how to start these services manually). Use `rpcinfo -p server` to see which services are currently running. You should have at least the following:

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100005    1   udp    960  mountd
 100005    3   udp    960  mountd
 100005    1   tcp    896  mountd
 100005    3   tcp    896  mountd
 100003    2   udp   2049  nfs
 100003    3   udp   2049  nfs
 100003    2   tcp   2049  nfs
 100003    3   tcp   2049  nfs
 100026    1   udp    978  bootparam
```

Note that some bootparam servers are somewhat sensitive. Some require fully qualified hostnames or partially qualified hostnames (which can be solved by having both fully and partially qualified entries). Other servers are case sensitive.

Another thing to check is if you are using `/etc/hosts` instead of DNS and you have an entry in `/etc/hosts` like

```
127.0.0.1    localhost server
```

This is to prevent self addressed packets from going via the physical network interface. It also means that `rpc.bootparamd` will send the wrong address (127.0.0.1) to the client. Run `rpc.bootparamd -d` to see the actual information sent to the client.

7. Build the swap file for myclient:

```
# mkdir /export/myclient
# cd /export/myclient
# dd if=/dev/zero of=swap bs=256k count=1024
```

   This creates a 256 Megabyte swap file.

8. Populate myclient's / filesystem on the server. How this is done depends on the client architecture and the version of the OpenBSD distribution. It can be as simple as copying and modifying the server's root filesystem, or perhaps you need to get those files out of the standard binary distribution.

9. Export the required filesystems in `/etc/exports`:

```
/export/myclient -maproot=root -alldirs myclient
```

   If the server and client are of the same architecture, then the client can share the server's `/usr` filesystem (as is done above). If not, you must build a (more or less) complete `/usr` partition for the client in some other place. If you have only one client (with a different architecture or operating system than the server), then you can use the client's root partition. However, if you have multiple clients, then it is more efficient to use a separate `/usr` partition, shared by all clients.

10. Copy and customize at least the following files in

```
/export/myclient/root:
    # cd /export/myclient/root/etc
    # cp /etc/hosts hosts
    # echo myclient > myname
```

Although the client already has acquired its network configuration from the boot process, it may be a good idea to add

```
echo "dhcp NONE NONE NONE" > hostname.xx0
```

where `xx0` is the name of the network interface the client used for booting. This will set additional information (e.g. `/etc/resolv.conf`). In general it is important not to have any network configuration information in the client partition to avoid inconsistencies in case you change the client configuration on the server.

11. Correct the critical mount points in the client's `/etc/fstab` (which will be `/export/myclient/root/etc/fstab`) i.e.,

    ```
    server:/export/myclient/root / nfs rw 0 0
    server:/usr /usr nfs ro 0 0
    ```

    The above lines must be there, you may also want to add additional partitions as well (e.g. `/home`).

    Note that there is no entry for a swap partition. You do not need an entry for the swap partition because this information is provided by `rpc.bootparamd`.