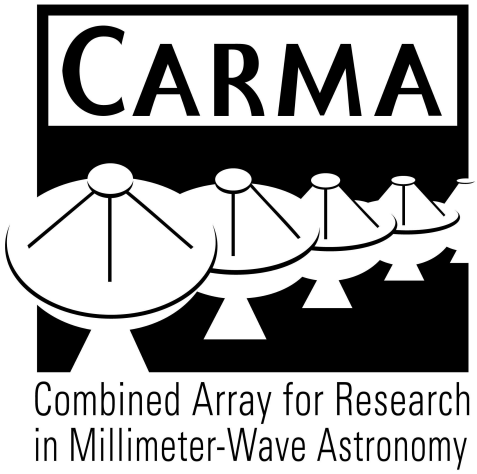

Computer Network Booting

Colby Kraybill, Radio Astronomy Lab, UC Berkeley

Andy Beard, OVRO, Caltech

Paul Daniel, OVRO, Caltech

Revision 1.6

05 December 2002

Change Record

REVISION	DATE	AUTHOR	SECTIONS/PAGES AFFECTED
	REMARKS		
1.4	10/25/02	Andy Beard	Mod 1.2, 1.2.2.1, 1.3.2.1 Add 1.2.2.3
	Added description of pxelinux bootloader and OVRO setup.		
1.5	10/25/02	Colby Kraybill	Mod 1.2
	Five stages instead of four now. Will need to rationalize pxe boot edits…		
1.6	12/05/02	Colby Kraybill	
	Pxe booting rationalized		

ABSTRACT

The antenna computers (*antputers*) on a standard CARMA antenna will not have any mass storage devices from which to boot or store data. Similarly, the line length measurement system and some elements of the COBRA correlator will have network booted computers. The operating system will have to be loaded across the network, as well as all software that runs on the computer to control and communicate with other systems. Fine details of how this is accomplished are discussed.

1.1 Design Decisions

The CARMA Computing Group has decided the following for Linux computer configurations:

- RedHat 7.3, 8.x
- Eliminate hard drive spindles where possible to reduce failure modes
- Linux Kernel 2.4.19
- All antputers will have a stable Ethernet connection.

In addition, this document assumes that the basic computer architecture (IA-32 running Linux RedHat 7.3) of the boot server is the same as the network booting computers. Network boot computers will most likely be in Compact PCI (cPCI) crates.

In this document, the term “network computer” is synonymous with “diskless machine”.

1.2 Booting Sequence

The computers that require network booting will complete this process in five stages. For this discussion, the cPCI computer attempting to boot is referred to as the remote computer.

Stage 1: The remote computer powers up and starts running Power On Self Tests. Once these tests complete, the computer sends broadcast queries to the local network, looking for a boot server. These queries do not require the remote computer to have an IP address.

Stage 2: A boot server listening for boot requests responds with information about the remote computer’s IP address, net mask, hostname, DNS server information, tftp server information and location information for either a network bootloader or network bootable kernel.

Stage 3: The remote computer sets it’s IP to the one sent back by the boot server and proceeds to attempt downloading the network bootloader or kernel from the location specified in stage 2.

Stage 4: If a bootloader was downloaded, it proceeds to download the kernel which can be generic and shared among multiple clients. It then passes kernel parameters to the kernel via a configuration file located on the boot server and specific to the clients IP address. Both the kernel and bootloader are loaded using the TFTP protocol. Once the kernel is loaded, the bootloader turns over control to the boot image.

Stage 5: The boot image (or kernel) begins loading like any normal Linux computer with one caveat: the kernel will use a remotely mounted (NFS) partition for its root partition (referred to as / or ‘slash’).

1.2.1 BIOS Support for Network Boot on a cPCI Computer (Stage 1)

The cPCI computer boards typically come with PreBoot Execution (PXE) support in BIOS ROMs. If the board you have does not support PXE, most likely you need a boot ROM upgrade.

> The current BIMA antputer boards (purchased from Ziatech, model 5531) required a BIOS update to version 5.11 for PXE support. A floppy image of this BIOS update and a utility to install the BIOS can be found here:
> http://celestial.berkeley.edu/CARMA/linux/diskless/
>
> You can use 'dd' to write this image to a floppy using the following command:
>
> % dd if=ziatechbiosfloppy.img of=/dev/floppy
>
> The location of your /dev entry for your floppy drive is system dependent.

The Ziatech 5531 BIOS should be set to boot off one of the Ethernet adapters using the PXE boot method. This should hold true for how to configure boards from other manufacturers.

There are no special parameters for PXE to work. The system uses the hardware Ethernet MAC address (XX:XX:XX:XX:XX) to query the local network for a boot server and the boot server responds with IP, net mask, hostname, DNS server and possibly a default gateway. In our system, the default gateway is unnecessary as it is a security risk to have network boot computers accessible outside of the local network.

1.2.2 Configuring the Boot Server (Stages 2 and 3)

1.2.2.1 DHCP

A standard RedHat installation does not include some of the utilities required to create and distribute network boot images. The first thing the boot server will have to be able to do is respond to PXE requests from remote computers. This can be accomplished using the DHCP server package that comes with the RedHat 7.3 CDs. DHCP is normally associated with just distributing IP's to laptops, but in this case it can also handle the PXE boot request.

On the first 7.3 CD:

os/i386/RedHat/RPMS/dhcp-2.0pl5-8.i386.rpm

On the first 8.0 CD:

Os/i386/RedHat/RPMS/dhcp-3.0pl1-9.i386.rpm

On the web:

http://celestial.berkeley.edu/linux/redhat/7.3/os/i386/RedHat/RPMS/dhcp-2.0pl5-8.i386.rpm

or

http://celestial.berkeley.edu/linux/redhat/8.0/os/i386/RedHat/RPMS/dhcp-3.0pl1-9.i386.rpm

Install this package using the rpm command (as root):

rpm –ivh dhcp-2.0pl5-8.i386.rpm

If successfully installed, you should now have a DHCP server configuration file in /etc called dhcp.conf. See the dhcp.conf man page for detailed information about the format of this file.

The DHCP daemon can be stopped, started and told to reload configuration information using the /etc/init.d/dhcpd script. To have the system start the daemon automatically at boot time, create a series of links from the directories in /etc/rc.d to point to /etc/init.d/dhcpd, for example, as root:

```
# ln –s /etc/init.d/dhcpd /etc/rc0.d/K35dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc1.d/K35dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc2.d/K35dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc3.d/S99dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc4.d/S99dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc5.d/S99dhcpd
# ln –s /etc/init.d/dhcpd /etc/rc6.d/K35dhcpd
```

The DHCP server's configuration file /etc/dhcp.conf will have entries that resemble the following:

```
host cobracpu1
{
        filename                        "pxelinux.0"
        next-server                     192.100.16.68
        hardware ethernet               00:80:42:10:ee:5b
        fixed-address                   192.100.16.180
        option vendor-class-identifier  "PXEClient"
        option vendor-encapsulated-options 01:04:00:00:00:00
}
```

"Filename" specifies the bootloader instead of a kernel. The next-server option is the IP address of the tftp server to be used. Thus the pxelinux.0 bootloader will be downloaded from the "next-server" in the directory specified by tftp options described below. The next-server can be the same as the dhcp server although this isn't necessary.

1.2.2.2 TFTP Server

Once the remote computer has received a proper response from the boot server, it can proceed to load the boot kernel from the boot server. This requires setting up a TFTP server on the boot server. TFTP is the Trivial File Transfer Protocol, which is primarily used for simple boot clients to load boot images across the network.

An RPM for the server can be found on the first RedHat 7.3 CD:

os/i386/RedHat/RPMS/tftp-server-0.28-2.i386.rpm

On the 8.0 CD:

os/i386/RedHat/RPMS/tftp-server-0.29-3.i386.rpm

Or on the web:

http://celestial.berkeley.edu/linux/redhat/7.3/os/i386/RedHat/RPMS/tftp-server-0.28-2.i386.rpm

or

http://celestial.berkeley.edu/linux/redhat/8.0/os/i386/RedHat/RPMS/tftp-server-0.29-3.i386.rpm

The TFTP daemon (tftpd) is loaded via xinetd (inetd on older Unixes) when a request comes in to the boot server for a connection to the tftp daemon. Under RedHat 7.3, the /etc/xinetd.d directory has all of the configuration files for the various servers it controls, including tftp. The contents of the /etc/xinetd.d/tftp file on the BIMA array boot server are:

```
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -u carma –s /carma/netboot
        disable                 = no
        per_source              = 11
        cps                     = 100 2
}
```

For an in depth explanation of these options, see the xinetd man page. The server_args are passed to tftpd and tell the server what user it should run as (for permissions) and where its root tree of files are, so that when the remote computer requests a file, those requests will start from that tree. In this specific case, during stage 1, the remote computer will have been given "/bima/ant1/kernel" (or "pxelinux.0" for bootloader configuration) as the file it needs to load, so the tftp daemon will attempt to return the contents of the file "/carma/antputers/bima/ant1/kernel".

1.2.2.3 PXELinux bootloader

PXELINUX is a PXE network bootloader included in the SYSLINUX package. Both are available on RedHat8.0 and should be available on 7.3 as well although it may not be included in the standard installation options. It is also available on the web at…

http://www.kernel.org/pub/linux/utils/boot/syslinux/RPMS/

or

http://celestial.berkeley.edu/linux/redhat/7.3/os/i386/RedHat/RPMS/syslinux-1.52-2.i386.rpm

or

http://celestial.berkeley.edu/linux/redhat/8.0/os/i386/RedHat/RPMS/syslinux-1.75-3.i386.rpm

After receiving its' IP address and other DHCP parameters from the boot server, the PXE client downloads the bootloader (named pxelinux.0) from the tftp server (specified in the dhcp "filename" and "next-server" options). This is done as well with tftp so the pxelinux.0 file must reside in the root tftp directory on the boot server.

Thus the pxelinux.0 bootloader resides at /carma/netboot/pxelinux.0

After the PXE client loads the bootloader from the server, the bootloader takes over the task of loading and configuring the kernel.

The bootloader then loads a configuration file from the tftp server based on its' IP address. The configuration files reside in a subdirectory of /carma/netboot called pxelinux.cfg. PXELinux first looks for a configuration file with the name of the hex IP address, if that is not found it looks for filenames with subsequent hex digits dropped from the IP address. If none are found it looks for a file named "default". For our 192.100.16.180 cobracpu1 (Hex address C06410B4) example it would look for the following files in order until one matching the name is found; "C06410B4", "C06410B", "C06410",…, "C", "default". In this manner, pxelinux enables one to specify configurations based on classes of IP addresses.

The configuration file itself has a form very similar to lilo.conf…

```
LABEL linux
        KERNEL bzImage
        APPEND root=/dev/nfs nfsroot=/carma/netboot/antputers/ovro/ant1,rw,nfsvers=2
```

The append command passes standard kernel "command line parameters" to the kernel. The root=/dev/nfs option tells the kernel to use an nfs mounted root filesystem. The nfsroot option tells the kernel where to get this from.

Multiple labels may be added such that different kernels or operating systems may be specified and selected by the PXE client. The kernel must reside in the default tftpboot directory as specified above (e.g. /carma/netboot/antputers/ovro/ant1/bzImage).

Once this is specified, pxelinux downloads the kernel and executes it with the specified parameters. The bootloaders' role is now over.

1.3 Compiling and Configuring Kernel for Network Booting

1.3.1 Kernel Build

To start from scratch, you must build the Linux kernel. Currently 2.4.19 is the blessed version for the CARMA project. A copy of the source can be found at either:

http://www.kernel.org

or

http://celestial.berkeley.edu/CARMA/linux/diskless/linux-2.4.19.tar.gz

The tar ball should be unpacked in /usr/src (it will create a directory called linux-2.4.19, you should make a symlink for /usr/src/linux -> /usr/src/linux-2.4.19 to make things easier).

Change into the /usr/src/linux directory and type:

```
# make menuconfig
```

Once the configuration menu comes up, select "Networking options", new selections will appear, look for "IP: kernel level autoconfiguration", select DHCP, BOOTP and RARP support.

Back out of the “Networking options” menu and then select “File Systems”, new selections will appear, look for “Network File Systems” and select it and more selections should appear. Select “NFS file system support”.

Be sure to make these selections as compiled into the kernel, not module support. Once you’ve done this, make any other configuration selections that you want and then exit the utility, it will ask you if you want to save the configuration.

Next you will build the kernel using the following commands:

```
# make –j3 dep
…
# make –j3 bzImage
…
# make modules
…
```

The –j3 argument will speed up the build by spawning up to three jobs during the build process. This has advantages on single-processor systems and even better results on multi-processor systems.

You should now have a bootable kernel. You can copy the kernel into the target root dir (see below). In the case of the boot server at Hat Creek, the kernel is copied into /carma/antputers/bima/ant1. At this point, it is a good idea to test everything set up so far. Try having the remote computer load and boot the kernel. It should be able to load the image and start the boot process, but will panic when it cannot load the NFS root directory (slash).

1.4 Root Tree (Stage 5)

Once the remote computer can load and execute the kernel, it is time to create a complete Linux system tree for the remote computer to load and complete the boot process.

During initial development, I decided to base everything off of a /carma tree. This is on the boot server but should also be a starting place for all CARMA related software on all Linux machines, including the trees for the network booted computers (i.e. they have their own /carma).

On the boot server, the following tree exists:

```
/carma
/carma/netboot
/carma/netboot/antputers
/carma/netboot/antputers/bima
/carma/netboot/antputers/bima/ant1
…
/carma/netboot/antputers/bima/antN

/carma/netboot/anputers/bima/dist/bin
/carma/netboot/antputers/bima/dist/etc
…
```

Once the remote computer has completed starting up the kernel, the kernel will attempt to NFS mount its root directory from /carma/antputers/bima/ant1 (note, this is also where the

kernel is loaded from). The antputer can later mount /carma/antputers/bima/dist as /carma locally. This can apply to any network booted computer for CARMA by simply creating a new branch, say, /carma/cobra/dist or /carma/linelength/dist, etc…

So to create the root tree for / on the BIMA antenna 1, do the following on the boot server:

(there will be a link to a script to automate these operations at some point)

```
# cd /carma/antputers/bima/ant1
# mkdir home mnt root tmp proc
# cp –a /var .
# cp –a /bin .
# cp –a /sbin .
# cp –a /boot .
# cp –a /usr .
# cp –a /lib .
# cp –a /etc .
# cp –a /dev .
# cd /usr/src/linux
# make modules_install INSTALL_MOD_PATH=/carma/antputers/bima/ant1
# cp System.map /carma/anputers/bima/ant1
# touch /carma/antputers/bima/ant1/fastboot
```

edit /carma/antputers/bima/ant1/fstab
(remove references to file system that will not exist on the remote computer)
(example from ant1's fstab file)

```
acc:/carma/anputers/bima/ant1 /          nfs     hard,intr          0 0
acc:/carma/anputers/bima/dist /carma     nfs     hard,intr          0 0
none                          /dev/pts   devpts  gid=5,mode=620     0 0
none                          /proc      proc    defaults           0 0
none                          /dev/shm   tmpfs   defaults           0 0
```

(in /carma/antputers/bima/ant1/etc)

```
# ln –s ../bin/true fsck.nfs
# rm fsck.ext2
# ln –s ../bin/true fsck.ext2
```

(in /carma/antputers/bima/ant1/etc/rc[2-6].d)

```
# rm K34yppasswdd K35dhcpd k35smb K35vncserver K72autofs K74ypserv
   K74ypxfrd K95kudzu S08ipchains S08iptables S09isdn S26apmd S60lpd
   S80sendmail S85gpm S90xfs S99nfs S05kudzu K75netfs
```

The start up files above are removed either because they are unnecessary for the operation of the remote computer or they interfere with startup/shutdown using an NFS mounted / partition.

(in /carma/antputers/bima/ant1/etc)

```
# rm logrotate.conf
# touch logrotate.conf
```

(in /carma/antputers/bima/ant1/etc)
edit syslog.conf, clear it out completely and leave the following lines:

```
*.*     /dev/console
*.*     @acc
```

All logging will be going through syslogd back to the ACC and not stored on the NFS mounted / partition (acc is the hostname of acc.astro.uiuc.edu).

The ACC's syslog daemon or whatever ends up being the main catcher for syslog info coming back from network booting computers should be run with the '-r' option.

Finally, the boot server must export these file systems, the BIMA boot server's /etc/exports file has the following:

```
/carma          192.17.14.0/24(rw,no_root_squash)
```

It is important to note that the no_root_squash option has been used to so that root on the remote computers can modify files on their NFS mounted / partitions.

1.5 Console Access

To circumvent the need for a VGA monitor and keyboard for these diskless machines, the BIOS can be configured to redirect console output to a serial port. The Ziatech boards allow for this as well as the Force Computer boards being used for the COBRA correlator.

At BIMA, we have been evaluating a product called the DS2-RPC from BayTech, Inc. These provide an Ethernet connection (for telnet access to control and monitor the power strip), four controllable power outlets and four serial ports. The serial ports can be connected to the serial port of the diskless computer, allowing a remote user to see what is being displayed on the console of the diskless machine. This also allows the remote user to watch the POST of the diskless machine and to reconfigure BIOS settings.

See: http://baytech.net:8080/cgi-private/product?product=DS2-RPC

1.6 Unresolved Issues

Rebooting requires using a hard reset after the normal shutdown programs have run, as the machine closes off its NFS mount and eth0 connection prematurely. The best way of going about fixing this doesn't seem straight forward to me as the normal process is buried across several shutdown files in /etc/rc.d

I am assuming there will be a private subnet for all diskless machines based on one of the blessed private network numbers (e.g. 10.0.x.x). The power strips can also be on this network. This information should be stored as part of the general array configuration information. The serial port access information should be incorporated into one of the antenna work packages (the DS2-RPC device specifically in the BIMA Antenna Misc. package and whatever device other members settle on).

Should create a script to automatically build a usable root tree for new installations or will a tar ball of a "template" suffice?

1.7 Appendix

Much of the information about diskless booting was gleaned from:

http://www.tldp.org/HOWTO/Network-boot-HOWTO

However, this document distills the information down into what I think is a more readable format and is directly relevant to the CARMA system.